УДК 681.326:519.713

В.И. Хаханов, д-р техн. наук, **И.В. Емельянов,** аспирант,
М.М. Любарский, аспирант, **С.В. Чумаченко,** д-р техн. наук,
Е.И. Литвинова, д-р техн. наук, **Тамер Бани Амер**, аспирант
Харьковский национальный университет радиоэлектроники
(Украина, 61166, Харьков, пр-т Ленина, 14,
тел. (057) 7021326, hahanov@icloud.com)

Кубитный метод дедуктивного анализа неисправностей для логических схем

Разработаны инновационные методы взятия булевых производных, синтеза тестов на их основе, а также дедуктивного моделирования неисправностей для функциональных элементов, заданных кубитными покрытиями. В методах анализа использованы векторные логические операции and, or, not, xor, а также операция встречного сдвига частей кубитной формы функциональности. Приведены примеры комбинационных схем для верификации и сравнительного анализа производительности базовых и предложенных методов. Описана структура встроенного процессора, выполняющего операции взятия производных, синтеза тестов, дедуктивного моделирования неисправностей для оценки качества проверяющих входных наборов и диагностирования. Предложенные технологии ориентированы на их имплементацию в виде облачного сервиса или IP инфраструктуры в архитектурах SoC.

К л ю ч е в ы е с л о в а: синтез тестов, проектирование и верификация SoC, кубитное покрытие, цифровая схема, моделирование неисправностей, булева производная, дедуктивное моделирование неисправностей.

Розроблено іноваційні методи взяття булевих похідних, синтезу тестів на їх основі, а також дедуктивного моделювання несправностей для функціональних елементів, заданих кубітними покриттями. В методах аналізу використано векторні логічні операції and, or, not, xor, а також операція зустрічного зсуву частин кубітної форми функціональності. Наведено приклади комбінаційних схем для верифікації та порівняльного аналізу продуктивності базових та запропонованих методів. Описано структуру вбудованого процесора, що виконує операції взяття похідних, синтезу тестів, дедуктивного моделювання несправностей для оцінки якості перевіряючих вхідних наборів і діагностування. Запропоновані технології орієнтовано на їх імплементацію у вигляді хмарного сервісу або IP інфраструктури в архітектурах SoC.

К л ю ч о в і с л о в а: синтез тестів, проектування і верифікація SoC, кубітне покриття, цифрова схема, моделювання несправностей, булева похідна, дедуктивне моделювання несправностей.

Вычисление булевых производных для Q-синтеза тестов. На рынке электроники наблюдается устойчивое доминирование инновационной киберкультуры, включающей наноаддитивные, биоинформационные, киберфизические, социальнокогнитивные компьютинговые технологии, создающие условия для киберфизического мониторинга и управления в значимых для человека областях, таких как образование, индустрия, транспорт, экология и высокое качество жизни. В рамках устойчивого развития киберфизического компьютинга интересными для рынка являются Quantum computing, Memory-Driven Computing, Cloud Computing, IoT Computing, Big Data Computing, создающие основу второй фазы компьютинга, на которой осуществляется активное управление всеми физическими процессами и явлениями (рис. 1).

Компания Hewlett Packard Enterprise создала прототип суперкомпьютера the Machine, использующий архитектуру memory-driven на фотонно-оптических соединениях, производительность которых на четыре порядка выше (в 8000 раз), чем существующих систем. Вычислительные процессы используют 8 терабайт памяти на мемристорах, что в 10 раз мощнее существующих серверных решений. Компания планирует использовать the Machine для решения проблем при анализе большого объема данных, например Transportation Systems, Smart Cities, Speech and Video Recognition, Security, Healthcare and IoT-computing. Таким образом, можно представить две парадигмы компьютинга, основанные на memorydriven технологиях, которые вписываются в структуру, представленную на рис. 2. Следует также интегрировать четыре самых существенных компонента, получивших устойчивое развитие на рынке электронных технологий, с которыми связано появление практически ориентированных треугольных взаимодействий (IoT, big data analytics, cyber physical systems), представленных на рис. 3.

Поскольку в ближайшие пять лет появятся квантовые вычислители, которые потребуют передела последовательной парадигмы алгоритмов на параллельные архитектуры, актуальной представляется эмуляция технологий параллельного синтеза и анализа на современных полупроводниковых компьютерах [1—3]. В рамках такой мотивации предлагаются кубитные структуры данных и методы их синтеза-анализа для memory-driven компьютинга, ориентированные на параллельное выполнение всех операций (рис. 4).

Представленные далее модели и методы являются логическим продолжением публикации [1], где изложены определения и структуры данных кубитного (квантового) компьютинга, а также метод синтеза тестов на основе анализа кубитного покрытия.

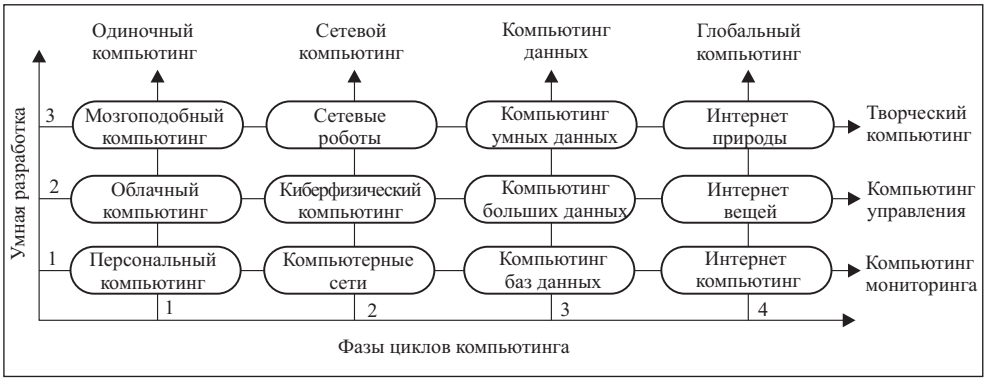


Рис. 1. Компьютинг во времени и пространстве

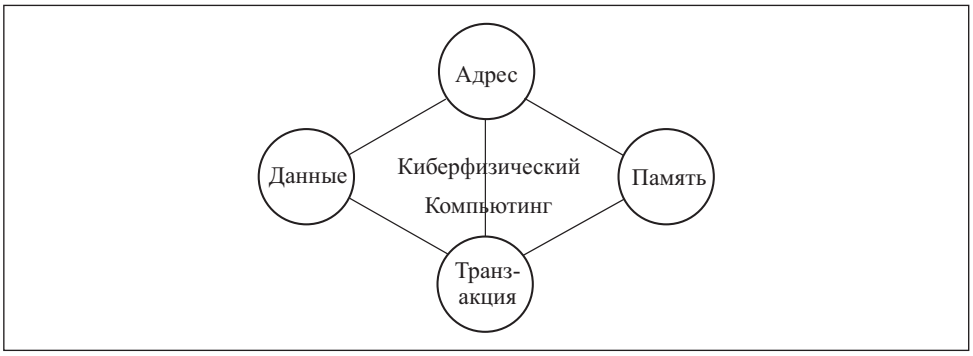


Рис. 2. Две технологии компьютинга

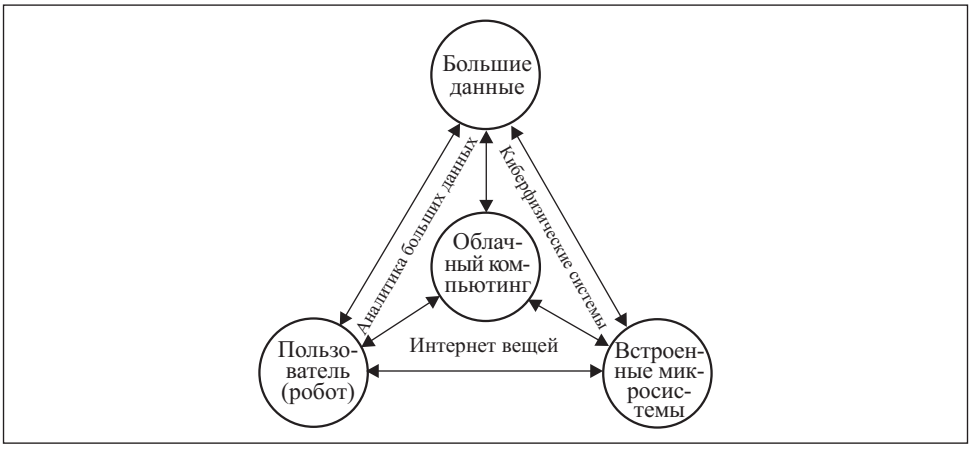


Рис. 3. Устойчивые компьютинговые триады рынка

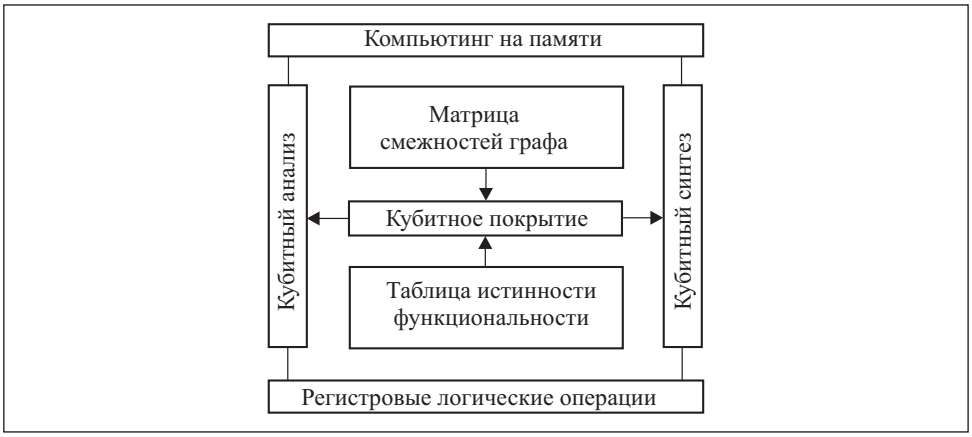


Рис. 4. Структура кубитного (квантового) компьютинга

Цель настоящего исследования — повышение производительности методов тестирования цифровых устройств, связанных с синтезом тестов и дедуктивным моделированием неисправностей на основе использования кубитной формы булевой производной.

Поставленные задачи:

1. Метод взятия булевой производной на основе кубитного или векторного представления логической функции.

2. Метод синтеза тестов на основе использования кубитной формы булевой производной.

3. Метод дедуктивного моделирования неисправностей на основе использования кубитной формы булевой производной.

4. Архитектура специализированного процессора встроенного тестирования на основе использования кубитной формы задания функциональности.

Рассмотрим метод взятия булевых производных по кубитному покрытию для создания условий активизации входных переменных при синтезе кубитных тестов. Проведем аналогию между двумя формами булевых функций для взятия производных: аналитической [4] и векторной. Исследование метода выполняем на примерах логических функций:

$$f(x)=x_1\vee x_1\bar{x}_2,\ f(x)=x_1x_2\vee\bar{x}_1x_3,\ f(x)=\bar{x}_2\bar{x}_3\vee x_1x_2x_3.$$

Вопросы, подлежащие решению:

1) определение производных первого порядка по аналитической и кубитной форме задания логической функции;

2) верификация полученных условий активизации путем их моделирования на одной из форм описания функциональности;

3) синтез тестов активизации переменных логической функции на основе вычисления производных.

Пример 1. Определить все производные первого порядка по кубитной форме логической функции $f(x) = x_1 \vee x_1 \bar{x}_2$. Формула вычисления по аналитическому выражению

$$\frac{df(x_1, x_2, ..., x_i, ..., x_n)}{dx_i} = f(x_1, x_2, ..., x_i = 0, ..., x_n) \oplus f(x_1, x_2, ..., x_i = 1, ..., x_n)$$

позволяет задать булеву производную первого порядка как сумму по модулю два нулевой и единичной остаточных функций. Для рассматриваемой функции получаем

$$\frac{df(x_1, x_2)}{dx_1} = f(0, x_2) \oplus f(1, x_2) = (0 \vee 0\bar{x}_2) \oplus (1 \vee 1\bar{x}_2) = 0 \oplus 1 = 1,$$

$$\frac{df(x_1, x_2)}{dx_2} = f(x_1, 0) \oplus f(x_1, 1) = (x_1 \vee x_1 \cdot \bar{0}) \oplus (x_1 \vee x_1 \cdot \bar{1}) =$$

$$= (x_1 \vee x_1 \cdot 1) \oplus (x_1 \vee x_1 \cdot 0) = (x_1 \vee x_1) \oplus (x_1 \vee 0) = x_1 \oplus x_1 = 0.$$

Нулевое значение производной означает отсутствие условий активизации для переменной x_2, что дает основания считать ее несущественной и, следовательно, убрать из числа переменных, формирующих функциональность. Далее предлагаются аналогичные преобразования для кубитного покрытия функции, заданного вектором:

X_1	X_2	Y
0	0	0
0	1	$0 = (0011)$.
1	0	1
1	1	1

Производную по таблице истинности можно вычислять посредством поочередного задания всех нулей и единиц в координатах столбцов, соответствующих каждой переменной:

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

Таким образом, производные по первой и второй переменной, записанные в формате кубитного покрытия, равны 1111 и 0000. Это означает,

что производная по первой переменной равна единице, а по второй — нулю. Однако такой результат можно получить более формально и технологично, не рассматривая входные наборы таблицы истинности, используя только логические операции встречного сдвига и последующей xor-операции над разрядами кубитного покрытия $\{a, b\} = a \oplus b$, где a, b — соседние подвекторы кубита $Q = (a, b)$:

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

Иначе для первой переменной необходимо xor-сложить сдвинутые одна относительно другой две половинки первого столбца, а результат записать в обе сдвигаемые симметричные области: $\{a, b\} = a \oplus b$, (11, 11) = 00 ⊕ 11. Для второй переменной следует рассматривать пары соседних координат столбца, а общий результат записывать в каждые сдвигаемые симметричные области-биты: $\{a, b\} = a \oplus b$: (0, 0) = 0 ⊕ 0 = 0, (0, 0) = 1 ⊕ 1 = 0. Таким образом, результат суммирования будет общим для каждой пары взаимодействующих подвекторов, размерность которых определяется номером рассматриваемой переменной от нуля до $2^n - 1$.

В общем случае формулу вычисления булевых производных для формирования соответствующей кубитной матрицы black box функциональности можно представить как xor-операцию над разрядами соседних групп битов:

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

Здесь n — число переменных в black box функциональности; i — номер переменной, по которой берется производная; $(j, j+1)$ — номера соседних групп в кубитном векторе, которые подлежат xor-сравнению, где число и мощность таких групп функционально зависит от номера переменной; $D_{(ij,\ ij+1)}(Q, X_i)$ — соседние группы производной по i-й переменной, формируемые xor-операцией. Под группой понимается совокупность битов, кратная степени двойки, подлежащая поразрядному xor-сложению с соответствующими битами соседней группы кубитного вектора.

Пример 2. Определить все производные первого порядка по аналитической форме логической функции $f(x) = x_1 x_2 \vee \bar{x}_1 x_3$. Для рассматриваемой функции выполняются следующие вычисления:

$$\frac{df(x_1,x_2,x_3)}{dx_1} = f(0,x_2,x_3) \oplus f(1,x_2,x_3) = (0 \cdot x_2 \vee \overline{0} \cdot x_3) \oplus (1 \cdot x_2 \vee \overline{1} \cdot x_3) =$$

$$= (0 \vee 1 \cdot x_3) \oplus (x_2 \vee 0 \cdot x_3) = x_3 \oplus x_2 = x_2\,\bar{x}_3 \vee \bar{x}_2\,x_3,$$

$$\frac{df(x_1,x_2,x_3)}{dx_2} = f(x_1,0,x_3) \oplus f(x_1,1,x_3) = \bar{x}_1 x_3 \oplus (x_1 \vee x_3) =$$

$$= \overline{\bar{x}_1 x_3}(x_1 \vee x_3) \vee \bar{x}_1 x_3 \overline{(x_1 \vee x_3)} = (x_1 \vee \bar{x}_3)(x_1 \vee x_3) \vee \bar{x}_1 x_3 \bar{x}_1 \bar{x}_3 = x_1,$$

$$\frac{df(x_1,x_2,x_3)}{dx_3} = f(x_1,x_2,0) \oplus f(x_1,x_2,1) = x_1 x_2 \oplus (\bar{x}_1 \vee x_2) =$$

$$= \overline{x_1 x_2}(\bar{x}_1 \vee x_2) \vee x_1\,x_2 \overline{(\bar{x}_1 \vee x_2)} = (\bar{x}_1 \vee \bar{x}_2)(\bar{x}_1 \vee x_2) \vee x_1 x_2 x_1 \bar{x}_2 = \bar{x}_1.$$

Для трех переменных получены четыре условия активизации, соответствующие четырем логическим путям в схемной структуре дизъюнктивной формы данной функции. Вычисление трех производных первого порядка по таблице истинности дает следующий результат:

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

Если исключить из рассмотрения таблицу истинности, а использовать кубитное покрытие, то на содержательном уровне процесс вычисления производных будет иметь следующий вид:

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

Пояснения. Для получения производной по первой переменной необходимо xor-сложить сдвинутые навстречу друг другу две соседние тетрады кубит-вектора Y, а результат записать в обе тетрады: $\{a,b\}=a\oplus b$, (0110, 0110) = 0101 ⊕ 0011. Для получения производной по второй переменной следует последовательно xor-сложить соседние пары кубит-вектора Y, а общий результат записывать в каждую пару: $\{a,b\}=a\oplus b$: (00, 00) = 01 ⊕ 01, (11, 11) = = 00 ⊕ 11. Для получения производной по третьей переменной следует последовательно xor-сложить соседние биты кубит-вектора Y, а общий результат записать в каждый соседний бит: $\{a,b\}=a\oplus b$: (1, 1) = 0 ⊕ 1, (1, 1) = 0 ⊕ 1, (0, 0) = 0 ⊕ 0, (0, 0) = 0 ⊕ 0. Естественно, что кубит-производная по любой входной переменной как вектор обладает относительной симметрией равенства подвекторов по построению: производная первой переменной имеет симметричное равенство двух тетрад, производная второй переменной имеет симметричное равенство каждых соседних пар, производная третьей переменной имеет симметричное равенство каждых соседних битов.

Пример 3. Определить все производные первого порядка по таблице истинности логической функции трех переменных: $f(x)=\bar{x}_2\bar{x}_3\vee x_1x_2x_3$. Результат взятия производных по таблице истинности [5] приведенной функциональности на основе выполнения операций над столбцами входных переменных представлен в следующем виде:

$$\frac{df}{dx_1}=$$

x_1	x_2	x_3	Y	Y_2^0	Y_2^1	$Y_2^{\oplus}$
0	0	0	1	1	1	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	1	0
1	0	1	0	0	0	0
1	1	0	0	0	0	0
1	1	1	1	0	1	1

$$=x_2x_3,$$

$$\frac{df}{dx_2}=$$

x_1	x_2	x_3	Y	Y_2^0	Y_2^1	$Y_2^{\oplus}$
0	0	0	1	1	0	1
0	0	1	0	0	0	0
0	1	0	0	1	0	1
0	1	1	0	0	0	0
1	0	0	1	1	0	1
1	0	1	0	0	1	1
1	1	0	0	1	0	1
1	1	1	1	0	1	1

$$=\bar{x}_1\bar{x}_3\vee x_1\bar{x}_3\vee x_1x_3=$$
$$=\bar{x}_3\vee x_1x_3,$$

$$\frac{df}{dx_3} =$$

x_1	x_2	x_3	Y	Y_3^0	Y_3^1	$Y_3^{\oplus}$
0	0	0	1	1	0	1
0	0	1	0	1	0	1
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	1	0	1
1	0	1	0	1	0	1
1	1	0	0	0	1	1
1	1	1	1	0	1	1

$$= \bar{x}_1\bar{x}_2 \vee x_1\bar{x}_2 \vee x_1x_2 = \bar{x}_2 \vee x_1x_2.$$

В качестве альтернативы представим результаты выполнения процедур взятия производных первого порядка для трех переменных по кубитному покрытию функциональности:

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

Процесс вычисления производных на кубитном покрытии без рассмотрения таблицы истинности на формальном уровне имеет следующий вид:

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

Интерпретация полученных кубит-производных. Естественно, что производные есть функции, заданные векторами. Они могут быть записаны в аналитической форме (дизъюнктивная нормальная форма (ДНФ)) по единичным значениям переменных, формирующих адреса ячеек кубит-вектора:

$$Y_1' = 011 \vee 111 = \overline{X}_1 X_2 X_3 \vee X_1 X_2 X_3 = X_2 X_3 (\overline{X}_1 \vee X_1) = X_2 X_3,$$
$$Y_2' = 000 \vee 010 \vee 100 \vee 101 \vee 110 \vee 111 = \overline{X}_1 \overline{X}_3 \vee X_1,$$
$$Y_3' = 000 \vee 001 \vee 100 \vee 101 \vee 110 \vee 111 = \overline{X}_1 \overline{X}_2 \vee X_1.$$

Минимизация булевых функций, соответствующих производным, приводит к аналитическим выражениям, где отсутствуют переменные, по которым берется производная. Таким образом, все результаты вычисления производных от трех форм (аналитическая, табличная, векторная) задания функции являются идентичными. Наиболее технологичен метод взятия производной по кубитному покрытию, имеющий меньшую вычислительную сложность в силу компактного представления функциональности. При использовании аналитической формы предполагается существенное повышение сложности алгоритмов, связанной с применением законов булевой алгебры и минимизации функций. Это ограничивает ее применение для решения практических задач.

Для сравнения кратко опишем алгоритм получения теста $T = [T_{ij}]$, $i = \overline{1,k}$; $j = \overline{1,n}$ комбинационной функциональности, заданной кубическим покрытием или таблицей истинности, содержащий следующие пункты [5]:

1. Вычисление производных по всем n переменным функциональности с использованием кубического покрытия,

$$f'(x_i) = f(x_1, x_2, ..., x_i = 0, ..., x_n) \oplus f(x_1, x_2, ..., x_i = 1, ..., x_n).$$

2. Объединение всех условий (векторов) активизации в таблицу, где каждому вектору с помощью конкатенации (*) ставится в соответствие изменение переменной, по которой была взята производная, что означает удвоение числа тестовых наборов по отношению к общему числу k условий активизации,

$$T = \bigcup_{i=1}^{n} [f'(x_i) * (x_i = 0) \vee (x_i = 1)].$$

3. Минимизация тестовых векторов путем удаления повторяющихся входных последовательностей,

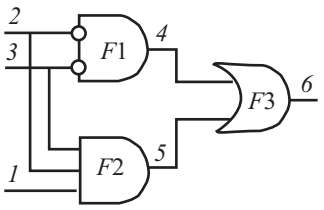
$$T_{ij} = T_{i-1,\,j} \leftarrow T_{ij} = X;\ T_{1j} = 1 \leftarrow T_{1j} = X.$$

4. $T = T \setminus T_i \leftarrow T_i = T_{i-r}$, $r = \overline{1, i-1}$, $i = \overline{2, n}$.

На рис. 5 представлены таблицы процесса получения теста в соответствии с пунктами 2, 3 алгоритма для функциональности $f(x) = \bar{x}_2 \bar{x}_3 \vee x_1 x_2 x_3$, представленной схемной структурой.

Метод синтеза тестов на основе кубитных булевых производных. В качестве альтернативы упомянутому выше алгоритму рассмотрим тех-

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. Тест для схемной структуры булевой функции: *1— 6* — линии схемы, *F*1 — *F*3 — логические элементы

нологически простой метод синтеза тестов на основе взятия производных по кубитным покрытиям функциональных элементов, без рассмотрения состояний входных переменных:

1. Исходное задание логической функциональности кубитным покрытием.

2. Выполнение операций встречного сдвига частей кубит-вектора и последующего покоординатного xor-суммирования для получения векторов производных для каждой входной переменной.

3. Логическое объединение векторов производных, формирующее тест-вектор, равный по размеру кубитному покрытию.

4. В случае необходимости получения минимального теста решается задача покрытия (уже на матрице кубит-производных) путем нахождения минимального числа пар единичных координат кубит-производных для всех переменных, где пара единиц должна проверять одиночные константные неисправности каждого входа. Процедура выбора пары единиц в кубит-производной определяется наличием двух представителей, по одному от любой четной и любой нечетной частей вектора.

В следующей таблице представлены результаты синтеза теста для логической функции от трех переменных, заданной уравнением $f(x) = \bar{x}_2\bar{x}_3 \vee x_1x_2x_3$, имеющей кубитное покрытие (10001001):

X_1	X_2	X_3	Y	Y_1'	Y_2'	Y_3'	T				
0	0	0	1	0	1	1	1	X_1	X_2	X_3	Y
0	0	1	0	0	0	1	1	0	1	1	0
0	1	0	0	0	1	0	1	1	1	1	1
0	1	1	0	1	0	0	1	0	1	0	0
1	0	0	1	0	1	1	0	0	0	0	1
1	0	1	0	0	1	1	0	0	0	1	0
1	1	0	0	0	1	1	0				
1	1	1	1	1	1	1	1				

При выполнении пункта 3 алгоритм синтеза дает минимальный тест проверки входных переменных, который содержит пять наборов, что представлено столбцом *T*, а также продублировано в явном виде таблицей справа.

Интерес представляет тот факт, что результат выполнения процедуры взятия производной по кубит-вектору уже содержит тест активизации каждой переменной. Объединенный тест проверяет все константные неисправности входных переменных, а также может быть использован для диагностирования неисправностей, поскольку для существенных входов все производные-векторы будут различными. Фактически взятие производной по переменной на кубит-покрытии формирует *Q*-тест.

Дедуктивный анализ неисправностей цифровых структур. Дедуктивные методы моделирования неисправностей для верификации тестов подробно рассмотрены в работах [5—9]. В описанных в них методах анализа качества тестов использованы аналитические и табличные формы задания функциональностей, обработка которых влияет на быстродействие. Предлагаемый метод моделирования неисправностей на основе использования кубитной формы описания функциональности практически ориентирован на обработку схем большой размерности и имеет линейную вычислительную сложность в зависимости от числа состояний входных переменных. Метод оперирует несколькими логическими регистровыми операциями (and, or, not, xor), которые выполняются параллельно, что обеспечивает высокое быстродействие вычислительных процессов моделирования неисправностей.

Дедуктивный анализ используется для определения качества теста относительно введенного класса неисправностей, как правило, одиночных константных. Существует развитая теория дедуктивного анализа [5], ориентированная на параллельную обработку списков неисправностей. Базовые понятия теории моделирования неисправностей представлены аппаратом транспортирования списков дефектов через примитивные функциональные логические элементы [6—12]. Далее определены дедуктивные функции параллельного моделирования неисправностей на исчерпывающем тесте для функциональных элементов and, or, not. Получение дедуктивного преобразователя для функции and:

$$L[T=(00,01,10,11),F=(X_1\wedge X_2)]=$$

$$=L\{(\bar{x}_1\bar{x}_2\vee\bar{x}_1x_2\vee x_1\bar{x}_2\vee x_1x_2)\wedge[(X_1\oplus T_{t1}\wedge X_2\oplus T_{t2})\oplus T_{t3}]\}=$$

$$=(\bar{x}_1\bar{x}_2)\{[(X_1\oplus 0)\wedge(X_2\oplus 0)]\oplus 0\}\vee(\bar{x}_1x_2)\{[(X_1\oplus 0)\wedge(X_2\oplus 1)]\oplus 0\}\vee$$

$$\vee(x_1\bar{x}_2)\{[(X_1\oplus 1)\wedge(X_2\oplus 0)]\oplus 0\}\vee(x_1x_2)\{[(X_1\oplus 1)\wedge(X_2\oplus 1)]\oplus 1\}=$$

$$=(\bar{x}_1\bar{x}_2)(X_1\wedge X_2)\vee(\bar{x}_1x_2)(X_1\wedge\overline{X}_2)\vee(x_1\bar{x}_2)(\overline{X}_1\wedge X_2)\vee(x_1x_2)(X_1\vee X_2).$$

Аналогично выполняются вычисления для функции or:

$$L[T=(00,01,10,11),F=(X_1\vee X_2)]=$$

$$=L\{(\bar{x}_1\bar{x}_2\vee\bar{x}_1x_2\vee x_1\bar{x}_2\vee x_1x_2)\wedge[(X_1\oplus T_{t1}\wedge X_2\oplus T_{t2})\oplus T_{t3}]\}=$$

$$=(\bar{x}_1\bar{x}_2)\{[(X_1\oplus 0)\vee(X_2\oplus 0)]\oplus 0\}\vee(\bar{x}_1x_2)\{[(X_1\oplus 0)\vee(X_2\oplus 1)]\oplus 1\}\vee$$

$$\vee(x_1\bar{x}_2)\{[(X_1\oplus 1)\vee(X_2\oplus 0)]\oplus 1\}\vee(x_1x_2)\{[(X_1\oplus 1)\vee(X_2\oplus 1)]\oplus 0\}=$$

$$=(\bar{x}_1\bar{x}_2)(X_1\vee X_2)\vee(\bar{x}_1x_2)(\bar{X}_1\wedge X_2)\vee(x_1\bar{x}_2)(X_1\wedge\bar{X}_2)\vee(x_1x_2)(X_1\wedge X_2).$$

Здесь $T_t=(T_{t1},T_{t2},T_{t3})$, $t=\overline{1,4}$ — тест-вектор, имеющий три координаты, и последняя определяет состояние выхода двухвходового элемента and (or); L — список выходных неисправностей; X — список неисправностей на конкретном входе примитива; $x=\{0,1\}$ — логическое значение на входе примитива. В преобразовании

$$L[T=(0,1),F=\bar{X}_1]=L\{(\bar{x}_1\vee x_1)[(\overline{X_1\oplus T_{t1}})\oplus T_{t2}]\}=$$

$$=\bar{x}_1[(\overline{X_1\oplus 0})\oplus 1]\vee x_1[(\overline{X_1\oplus 1})\oplus 0]=\bar{x}_1\bar{\bar{X}}_1\vee x_1\bar{\bar{X}}_1=\bar{x}_1X_1\vee x_1X_1$$

$T_t=(T_{t1},T_{t2})$, $t=\overline{1,2}$ — тест-вектор, имеющий две координаты, где вторая — состояние выхода инвертора.

Последнее выражение иллюстрирует инвариантность инверсии ко входному набору для транспортирования дефектов. Она трансформируется в повторитель. Поэтому данная функция не фигурирует на выходах дедуктивных элементов.

Совместная аппаратурная реализация дедуктивных функций для оставшихся двухвходовых элементов and, or на исчерпывающем тесте представлена универсальной схемой дедуктивно-параллельного анализа неисправностей (рис. 6). В симуляторе представлены булевы ($x1$, $x2$) и регистровые ($X1$, $X2$) для кодирования неисправностей входы, переменная V выбора типа исправной функции (and, or), выходная регистровая переменная Y. Состояния двоичных входов $x1$, $x2$ и переменная выбора элемента определяют одну из четырех дедуктивных функций для получения вектора Y проверяемых неисправностей.

Предлагаемая технологическая реализация дедуктивного моделирования по кубитной форме задания функциональностей отличается от описанной выше параллелизмом выполнения логических операций, а также возможностью применения метода для любых цифровых структур. Совокупность кубит-производных для всех входных переменных, вычисленных по кубитному покрытию, представляет собой кубитную матрицу для реализации дедуктивного метода моделирования неисправностей. Строка матрицы формирует условия для транспортирования списков неисправностей от внешних входов до выхода по следующему правилу: единичные

значения создают объединение входных списков, а нулевые сигналы указывают на входы, списки которых должны быть вычтены из результата объединения. Наличие всех нулевых сигналов в строке создает условия пересечения входных списков между собой.

Пример 4. Предлагается построение дедуктивных формул транспортирования списков неисправностей от входных переменных до выхода функциональности $f(x)=\bar{x}_2\bar{x}_3 \vee x_1x_2x_3$, заданной входными наборами, кубитным покрытием с векторными производными:

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

Для повышения быстродействия моделирования неисправностей необходимо исключить аналитические выражения, чтобы оперировать только кубитными векторами. При этом следует учитывать жесткое ограничение на присутствие в схеме только одиночных константных неисправностей, которые не могут параллельно приходить на несколько входов функционального модуля. В данном случае сокращенная и развернутая формулы дедуктивного моделирования одиночных константных неисправностей для логических функциональностей, представленных в виде кубитных векторов, имеют следующий вид:

$$L=\bigvee_{i=1}^{2^n}\{x_i \wedge [L(X_i^1)\wedge \bar{L}(X_i^0)]\},$$

$$L=\bigvee_{i=1}^{2^n}\{(x_{i1}, x_{i2}, ..., x_{i2}, ..., x_{in})\wedge [\bigvee_{X_{ij}=1} L(X_{ij}) \bigwedge_{X_{ij}=0} \bar{L}(X_{ij})]\}.$$

Здесь L — список выходных неисправностей; $L(X)$ — списки или векторы неисправностей на входах; x — матрица входных тестовых наборов; X — матрица кубитных производных от кубитного покрытия; n — число входных переменных.

А л г о р и т м построения дедуктивной формулы для заданной функциональности:

1. Задание кубит-вектора функциональности.

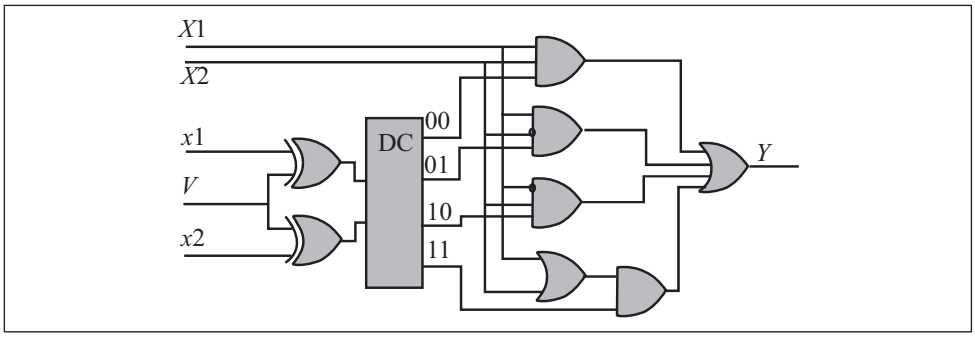


Рис. 6. Симулятор неисправностей примитивов

2. Вычисление кубит-производных для входных переменных в целях получения соответствующей матрицы.

3. Формирование аналитической или матрично-векторной формы вычисления выходных списков неисправностей путем логического умножения матриц входных тестовых воздействий и матрицы производных.

Процесс вычисления аналитической и векторной форм для дедуктивного моделирования неисправностей логической функциональности имеет вид

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

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

$$L=(000\wedge 011)\vee(001\wedge 001)\vee(010\wedge 010)\vee(011\wedge 100)\vee$$

$$\vee(100\wedge 011)\vee(101\wedge 011)\vee(110\wedge 011)\vee(111\wedge 111),$$

$$L=(\bar{x}_1\bar{x}_2\bar{x}_3\wedge\overline{X}_1X_2X_3)\vee(\bar{x}_1\bar{x}_2x_3\wedge(\overline{X}_1\vee\overline{X}_2)X_3)\vee(\bar{x}_1x_2\bar{x}_3\wedge(\overline{X}_1\vee\overline{X}_3)X_2)\vee$$

$$\vee(\bar{x}_1x_2x_3\wedge X_1(\overline{X}_2\vee\overline{X}_3))\vee(x_1\bar{x}_2\bar{x}_3\wedge\overline{X}_1X_2X_3)\vee(x_1\bar{x}_2x_3\wedge\overline{X}_1X_2X_3)\vee$$

$$\vee(x_1x_2\bar{x}_3\wedge\overline{X}_1X_2X_3)\vee(x_1x_2x_3\wedge X_1X_2X_3).$$

Технологическая сложность предложенного метода представлена в виде процедуры получения дедуктивных формул моделирования неисправностей [5] на основе аналитического задания функциональностей:

$$L=\bar{x}_1\bar{x}_2\bar{x}_3\{[(\overline{X}_2\oplus 0)(\overline{X}_3\oplus 0)\vee(X_1\oplus 0)(X_2\oplus 0)(X_3\oplus 0)]\oplus 1\}\vee$$

$$\vee\bar{x}_1\bar{x}_2x_3\{[(\overline{X}_2\oplus 0)(\overline{X}_3\oplus 1)\vee(X_1\oplus 0)(X_2\oplus 0)(X_3\oplus 1)]\oplus 0\}\vee$$

$$\vee\bar{x}_1x_2\bar{x}_3\{[(\overline{X}_2\oplus 1)(\overline{X}_3\oplus 0)\vee(X_1\oplus 0)(X_2\oplus 1)(X_3\oplus 0)]\oplus 0\}\vee$$

$$\vee\bar{x}_1x_2x_3\{[(\overline{X}_2\oplus 1)(\overline{X}_3\oplus 1)\vee(X_1\oplus 0)(X_2\oplus 1)(X_3\oplus 1)]\oplus 0\}\vee$$

$$\vee x_1\bar{x}_2\bar{x}_3\{[(\overline{X}_2\oplus 0)(\overline{X}_3\oplus 0)\vee(X_1\oplus 1)(X_2\oplus 0)(X_3\oplus 0)]\oplus 1\}\vee$$

$$\vee x_1\bar{x}_2x_3\{[(\overline{X}_2\oplus 0)(\overline{X}_3\oplus 1)\vee(X_1\oplus 1)(X_2\oplus 0)(X_3\oplus 1)]\oplus 0\}\vee$$

$$\vee x_1x_2\bar{x}_3\{[(\overline{X}_2\oplus 1)(\overline{X}_3\oplus 0)\vee(X_1\oplus 1)(X_2\oplus 1)(X_3\oplus 0)]\oplus 0\}\vee$$

$$\vee x_1x_2x_3\{[(\overline{X}_2\oplus 1)(\overline{X}_3\oplus 1)\vee(X_1\oplus 1)(X_2\oplus 1)(X_3\oplus 1)]\oplus 1\},$$

$$L=\bar{x}_1\bar{x}_2\bar{x}_3(\overline{\overline{X}_2\overline{X}_3\vee X_1X_2X_3})\vee\bar{x}_1\bar{x}_2x_3(\overline{X}_2X_3\vee X_1X_2\overline{X}_3)\vee$$

$$\vee\bar{x}_1x_2\bar{x}_3(X_2\overline{X}_3\vee X_1\overline{X}_2X_3)\vee\bar{x}_1x_2x_3(X_2X_3\vee X_1\overline{X}_2\overline{X}_3)\vee$$

$$\vee x_1\bar{x}_2\bar{x}_3(\overline{\overline{X}_2\overline{X}_3\vee\overline{X}_1X_2X_3})\vee x_1\bar{x}_2x_3(\overline{X}_2X_3\vee\overline{X}_1X_2\overline{X}_3)\vee$$

$$\vee x_1x_2\bar{x}_3(X_2\overline{X}_3\vee\overline{X}_1\overline{X}_2X_3)\vee x_1x_2x_3(\overline{\overline{X_2X_3}\vee\overline{X}_1\overline{X}_2\overline{X}_3}),$$

$$L=\bar{x}_1\bar{x}_2\bar{x}_3(\overline{\overline{X}_2\overline{X}_3}\wedge\overline{X_1X_2X_3})\vee\bar{x}_1\bar{x}_2x_3(\overline{X}_2X_3\vee X_1X_2\overline{X}_3)\vee$$

$$\vee\bar{x}_1x_2\bar{x}_3(X_2\overline{X}_3\vee X_1\overline{X}_2X_3)\vee\bar{x}_1x_2x_3(X_2X_3\vee X_1\overline{X}_2\overline{X}_3)\vee$$

$$\vee x_1\bar{x}_2\bar{x}_3(\overline{\overline{X}_2\overline{X}_3}\wedge\overline{\overline{X}_1X_2X_3})\vee x_1\bar{x}_2x_3(\overline{X}_2X_3\vee\overline{X}_1X_2\overline{X}_3)\vee$$

$$\vee x_1x_2\bar{x}_3(X_2\overline{X}_3\vee\overline{X}_1\overline{X}_2X_3)\vee x_1x_2x_3(\overline{X_2X_3}\wedge\overline{\overline{X}_1\overline{X}_2\overline{X}_3}),$$

$$L=\bar{x}_1\bar{x}_2\bar{x}_3[(X_2\vee X_3)\wedge(\overline{X}_1\vee\overline{X}_2\vee\overline{X}_3)]\vee\bar{x}_1\bar{x}_2x_3(\overline{X}_2X_3\vee X_1X_2\overline{X}_3\vee$$

$$\vee\bar{x}_1x_2\bar{x}_3(X_2\overline{X}_3\vee X_1\overline{X}_2X_3)\vee\bar{x}_1x_2x_3(X_2X_3\vee X_1\overline{X}_2\overline{X}_3)\vee$$

$$\vee x_1\bar{x}_2\bar{x}_3[(X_2\vee X_3)\wedge X_1\vee\overline{X}_2\vee\overline{X}_3)]\vee x_1\bar{x}_2x_3(\overline{X}_2X_3\vee\overline{X}_1X_2\overline{X}_3)\vee$$

$$\vee x_1x_2\bar{x}_3(X_2\overline{X}_3\vee\overline{X}_1\overline{X}_2X_3)\vee x_1x_2x_3[(\overline{X}_2\vee\overline{X}_3)\wedge X_1\vee X_2\vee X_3),$$

$$L=\bar{x}_1\bar{x}_2\bar{x}_3(\overline{X}_1X_2\vee\overline{X}_1X_3\vee\overline{X}_2X_3\vee X_2\overline{X}_3)\vee\bar{x}_1\bar{x}_2x_3(\overline{X}_2X_3\vee X_1X_2\overline{X}_3)\vee$$

$$\vee\bar{x}_1x_2\bar{x}_3(X_2\overline{X}_3\vee X_1\overline{X}_2X_3)\vee\bar{x}_1x_2x_3(X_2X_3\vee X_1\overline{X}_2\overline{X}_3)\vee$$

$$\vee x_1\bar{x}_2\bar{x}_3(X_1X_2\vee X_1X_3\vee\overline{X}_2X_3\vee X_2\overline{X}_3)\vee x_1\bar{x}_2x_3(\overline{X}_2X_3\vee\overline{X}_1X_2\overline{X}_3)\vee$$

$$\vee x_1x_2\bar{x}_3(X_2\overline{X}_3\vee\overline{X}_1\overline{X}_2X_3)\vee x_1x_2x_3(X_1\overline{X}_2\vee X_1\overline{X}_3\wedge X_2\overline{X}_3\vee\overline{X}_2X_3).$$

Вычисление кубитных форм для дедуктивного моделирования неисправностей основных логических примитивов or, and, xor представим в виде таблицы:

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

Здесь определены два вектор-столбца табличного задания входных переменных (x_1, x_2), кубит-вектор функции or — $Y^{\vee}$, две производные ($X_1^{\vee}$, $X_2^{\vee}$) для каждой входной переменной; показаны кубит-вектор задания функции and и два столбца производных, а также кубит-вектор функции xor и два столбца производных по входным переменным.

Формулы дедуктивного моделирования тривиально записываются по строкам таблицы:

$$L^{\vee} = \bar{x}_1\bar{x}_2(X_1 \vee X_2) \vee \bar{x}_1 x_2(\overline{X}_1 \wedge X_2) \vee x_1\bar{x}_2(X_1 \wedge \overline{X}_2) \vee x_1 x_2(X_1 \wedge X_2),$$

$$L^{\wedge} = \bar{x}_1\bar{x}_2(X_1 \wedge X_2) \vee \bar{x}_1 x_2(X_1 \wedge \overline{X}_2) \vee x_1\bar{x}_2(\overline{X}_1 \wedge X_2) \vee x_1 x_2(X_1 \vee X_2),$$

$$L^{\oplus} = \bar{x}_1\bar{x}_2(X_1 \vee X_2) \vee \bar{x}_1 x_2(X_1 \vee X_2) \vee x_1\bar{x}_2(\overline{X}_1 \vee X_2) \vee x_1 x_2(X_1 \vee X_2) =$$

$$= (X_1 \vee X_2).$$

Здесь входные переменные x_i соединены знаком коньюнкции, а производные — переменные транспортирования списков входных неисправностей, обозначенные символом X_i, соединены знаком дизъюнкции, в соответствии с состояниями координат столбцов-производных.

Таким образом, предложенная технология моделирования неисправностей, основанная на использовании векторных кубитных форм задания функциональностей и производных, не имеет аналогов по доступности понимания, простоте реализации и быстродействию. Чтобы получить дедуктивные формулы транспортирования дефектов любой функциональности, необходимо и достаточно получить вектор-столбцы производных всех входных переменных с помощью операций сдвига и xor-суммирования. Вычислительная сложность данных операций зависит от аппаратных затрат и может быть сведена к линейной зависимости от числа входных переменных.

На рис. 7 представлен процессор кубитного моделирования цифровых устройств, в который входят следующие структуры: исправного интерпретативного моделирования, дедуктивного анализа неисправностей, предназначенного для оценки качества теста и построения таблицы неисправ-

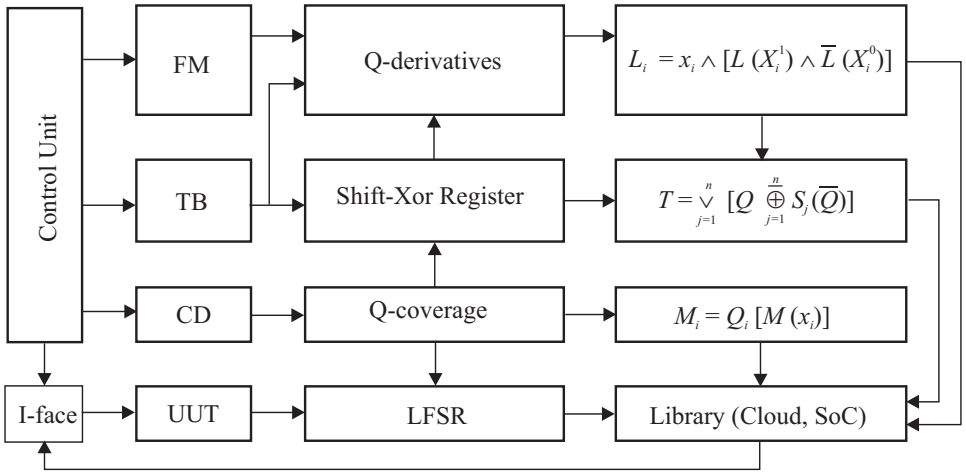
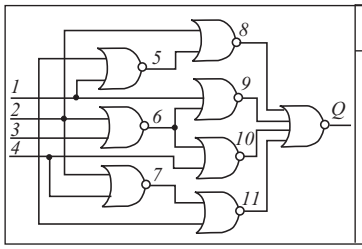


Рис. 7. Процессор кубитного моделирования цифровых устройств: Control Unit — устройство управления симулятором, синхронизирующее работу блоков исправного моделирования и структурных компонентов дедуктивного анализа неисправностей; FM — Fault Matrix, матрица входных неисправностей рассматриваемой функциональности цифрового устройства; TB — Test Bench, упорядоченная совокупность входных проверяющих последовательностей, где текущий входной набор идентифицируется как x_i; CD — Circuit Description, схемное описание цифрового устройства, где функциональные элементы представлены кубитными покрытиями Q-coverage; LFSR — регистр сдвига

ностей, а также модулей тестирования и диагностирования дефектов на стадиях проектирования и эксплуатации. Основное отличие от существующих решений заключается в использовании Q-покрытия, представленного в форме вектора состояний функциональности, что дает возможность существенно повысить быстродействие моделирования посредством выполнения параллельных регистровых операций.

Обработка кубитных покрытий осуществляется блоком исправного моделирования $M_i = Q_i[M(x_i)]$, реализующим адресные транзакции между кубитным покрытием и вектором моделирования M. Результаты исправного моделирования входных наборов формируют матрицу GM (Good simulation Matrix), записываемую в Library. Блок Shift-Xor Register формирует матрицу производных в кубитной форме (Q-derivatives), применяя регистровые операции сдвига и xor. В L-блоке для создания выходного списка неисправностей используется формула объединения списков неисправностей по всем единичным координатам строки матрицы производных с последующим вычитанием пересечения списков дефектов, соответствующих нулевым координатам: $L_i = x_i \wedge [L(X_i^1) \wedge \overline{L}(X_i^0)]$.

По результатам дедуктивного анализа формируются выходные списки неисправностей, соответствующие входным наборам, которые объеди-



Q	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
X_1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
X_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
X_3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
X_4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Рис. 8. Схема Шнейдера и таблица истинности: *1—11* — линии схемы

няются в DM (Fault Detected Matrix) и заносятся в Library. Кроме того, в *T*-модуле формируется *Q*-тест, оценивающий его качество в метрике одиночных константных неисправностей внешних входов и выходов функциональных элементов, которые заносятся в Library. Тесты для функциональностей вместе с матрицами неисправностей формируют библиотеку Library ={Signature, *Q*-coverage, *Q*-test, Quality, DM, GM}, которая может многократно использоваться как облачный или встроенный в SoC сервис для тестирования и(или) диагностирования функциональностей UUT (Unit Under Test) на основе использования интерфейса *I*-face, поддерживающего стандарты IEEE 1500 SECT, IP (Internet Protocol).

Поиск тестовых сервисов в библиотеке осуществляется по кубитному вектору, предварительно свернутому в 16-разрядный двоичный код — сигнатуру (Sign), на основе регистра сдвига с линейными обратными связями (LFSR), что дает возможность структурировать библиотеку для быстрого извлечения данных и проведения тестирования в режиме online. Блок LFSR выполняет две функции сжатия в 16-ричный код-сигнатуру: *Q*-покрытия и реакции UUT на входные тестовые воздействия для последующего диагностирования места, причины и вида дефекта.

Анализ схемы Шнейдера. Для того чтобы показать технологичность и производительность методов и алгоритмов, использующих кубитные покрытия функциональных примитивов, представим синтез тестов и дедуктивных формул моделирования неисправностей на основе булевых производных по входным переменным для схемы Шнейдера, изначально описанной таблицей истинности и *Q*-покрытием (рис. 8). Упрощенная формула определения производных Q' по входным переменным оперирует xor-взаимодействием (рис. 9) пар соседних компонентов кубитного покрытия, которое в данном случае имеет вид (1000000000000001):

$$Q'(X)=\{Q_i^L, Q_i^R\}=Q_i^L \oplus Q_i^R,\ i=\overline{1,2^{k-1}},\ k=\overline{1,n}.$$

Мощность каждого компонента, как число битов *i*, зависит от номера рассматриваемой переменной *k*, т.е. соседними парами могут быть биты,

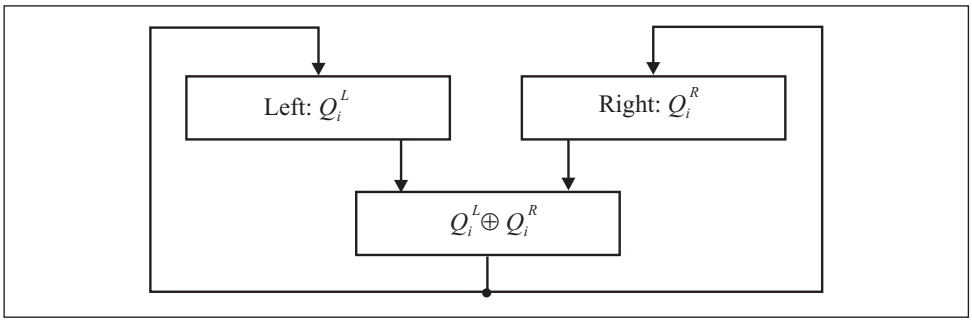


Рис. 9. Схема взятия Q-производной

двойки, четверки, восьмерки битов. Общее число переменных задано номером n. Его увеличение приводит к возрастанию разрядности соседних компонентов и соответствующему уменьшению их общего числа, необходимого для покрытия кубитного вектора.

Для получения производной по первой переменной схемы Шнейдера необходимо последовательно xor-сложить соседние биты в каждой паре кубит- вектора Q, а результат записать в каждый бит пары по правилу

$$(a,b) = a \oplus b\text{: } \{1, 1\} = 1 \oplus 0, (0, 0) = 0 \oplus 0, (0, 0) = 0 \oplus 0, (0, 0) = 0 \oplus 0,$$
$$(0, 0) = 0 \oplus 0, (0, 0) = 0 \oplus 0, (0, 0) = 0 \oplus 0, (1, 1) = 0 \oplus 1.$$

Для получения производной по второй переменной необходимо последовательно xor-сложить соседние пары кубит-вектора Q, а результат записать в каждую пару:

$$(a,b) = a \oplus b\text{: } \{10, 10\} = 10 \oplus 00, (00, 00) = 00 \oplus 00,$$
$$(00, 00) = 00 \oplus 00, (01, 01) = 00 \oplus 01.$$

Для получения производной по третьей переменной необходимо последовательно xor-сложить пары соседних тетрад кубит-вектора Q, а результат записать в обе тетрады:

$$\{a,b\} = a \oplus b\text{: } \{1000, 1000\} = 1000 \oplus 0000, (0001, 0001) = 0000 \oplus 0001.$$

Для получения производной по четвертой переменной следует xor-сложить две соседние восьмерки битов кубит-вектора Q, а результат записать в обе восьмерки битов:

$$\{a,b\} = a \oplus b\text{: } \{10000001, 10000001\} = 10000000 \oplus 00000001.$$

Табл. 1 содержит Q-покрытие и четыре вектора, представляющих собой производные по четырем переменным. Аналитическая форма производных, записанная по единичным значениям кубитного покрытия в формате входных переменных имеет следующий вид:

$$Q'(X_1) = \overline{X}_1\overline{X}_2\overline{X}_3\overline{X}_4 \vee X_1\overline{X}_2\overline{X}_3\overline{X}_4 \vee \overline{X}_1X_2X_3X_4 \vee X_1X_2X_3X_4 = \\ = \overline{X}_2\overline{X}_3\overline{X}_4 \vee X_2X_3X_4,$$

$$Q'(X_2) = \overline{X}_1\overline{X}_2\overline{X}_3\overline{X}_4 \vee \overline{X}_1X_2\overline{X}_3\overline{X}_4 \vee X_1\overline{X}_2X_3X_4 \vee X_1X_2X_3X_4 = \\ = \overline{X}_1\overline{X}_3\overline{X}_4 \vee X_1X_3X_4,$$

$$Q'(X_3) = \overline{X}_1\overline{X}_2\overline{X}_3\overline{X}_4 \vee \overline{X}_1\overline{X}_2X_3\overline{X}_4 \vee X_1X_2\overline{X}_3X_4 \vee X_1X_2X_3X_4 = \\ = \overline{X}_1\overline{X}_2\overline{X}_4 \vee X_1X_2X_4,$$

$$Q'(X_4) = \overline{X}_1\overline{X}_2\overline{X}_3\overline{X}_4 \vee \overline{X}_1\overline{X}_2\overline{X}_3X_4 \vee X_1X_2X_3\overline{X}_4 \vee X_1X_2X_3X_4 = \\ = \overline{X}_1\overline{X}_2\overline{X}_3 \vee X_1X_2X_3.$$

Процедура синтеза теста для схемы Шнейдера на основе производных. Каждая полученная функция представляет собой условия активизации переменной, по которой берется производная. Это означает, что если на каждый вход подать в двух временных тактах изменения сигналов 01 или 10, то в совокупности получим тест, проверяющий константные неисправности на всех логических путях, задаваемых условиями активизации. Применительно к схеме Шнейдера, такой тест будет содержать 16 входных наборов. Минимальный тест в формате переменных ($X_1X_2X_3X_4$), который не содержит повторяющихся входных наборов, имеет всего 10 векторов, записанных в правом столбце табл. 2.

Результат синтеза теста, полученного с помощью взятия производных в формате Q-вектора, имеет вид

$$T(A) = (1110100110010111).$$

Таблица 1

$Q'(X)$	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
X_1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
X_2	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1
X_3	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1
X_4	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1

Для синтеза Q-тестов на основе встречного S-сдвига компонентов кубитного покрытия используем следующую формулу:

$$T(S) = \bigvee_{j=1}^{n} \left[Q \overline{\bigoplus_{j=1}^{n}} S_j(\overline{Q}) \right].$$

Результат использования формулы, включающей выполнение четырех логических регистровых операций, представлен в табл. 3.

Таким образом, оба метода синтеза тестов (на основе встречного сдвига и активизации) дали одинаковый результат, содержащий по 10 входных векторов, которые проверяют все логические пути в схеме и все одиночные константные неисправности линий:

$$T(S) = T(A) = (1110100110010111).$$

Однако тест можно получить более просто, использовав параллельную логическую векторную операцию дизъюнкции над координатами Q-векторов производных, по правилу

$$T(Q') = \bigvee_{i=1}^{n} Q'(X_i).$$

Таблица 2

Activation	Test	Min
$\overline{X}_2\overline{X}_3\overline{X}_4 \vee X_2X_3X_4$	0000	0000
	1000	1000
	0111	0111
	1111	1111
$\overline{X}_1\overline{X}_3\overline{X}_4 \vee X_1X_3X_4$	0000	
	0100	0100
	1011	1011
	1111	
$\overline{X}_1\overline{X}_2\overline{X}_4 \vee X_1X_2X_4$	0000	
	0010	0010
	1101	1101
	1111	
$\overline{X}_1\overline{X}_2\overline{X}_3 \vee X_1X_2X_3$	0000	
	0001	0001
	1110	1110
	1111	

Результат выполнения регистровой операции дизъюнкции представлен в нижней строке табл. 4.

Таким образом, представленные три метода синтеза тестов на основе анализа кубитного покрытия функциональности, имеющие различные вычислительные сложности, дают одинаковый результат:

$$T(Q') = T(S) = T(A) = (1111010011001011 1).$$

В результате дальнейшей минимизации теста на основе моделирования получаем всего шесть наборов, которые представлены в табл. 5 и 6 соответственно исправного моделирования и анализа неисправностей.

Кубитно-дедуктивный метод анализа неисправностей. Аналитическая интерпретация таблицы кубитных производных $Q'(X)$, повернутая на 90 градусов для написания дедуктивных формул и последующего моделирования неисправностей, также проста для понимания. Практически

Таблица 3

Q — Test Synthesis	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
$\overline{Q}$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
$S_1(\overline{Q})$	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1
$S_2(\overline{Q})$	1	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1
$S_3(\overline{Q})$	1	1	1	1	0	1	1	1	1	1	1	0	1	1	1	1
$S_4(\overline{Q})$	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1
$T_1 = Q \overline{\oplus} S_1(\overline{Q})$	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
$T_2 = Q \overline{\oplus} S_2(\overline{Q})$	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1
$T_3 = Q \overline{\oplus} S_3(\overline{Q})$	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1
$T_4 = Q \overline{\oplus} S_4(\overline{Q})$	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
$T(S) = T_1 \vee T_2 \vee T_3 \vee T_4$	1	1	1	0	1	0	0	1	1	0	0	1	0	1	1	1

Таблица 4

$Q'(X)$	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
$Q'(X_1)$	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
$Q'(X_2)$	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1
$Q'(X_3)$	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1
$Q'(X_4)$	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
$T(Q') = \bigvee_{i=1}^{n} Q'(X_i)$	1	1	1	0	1	0	0	1	1	0	0	1	0	1	1	1

процедура формирования выходного списка (вектора) транспортируемых одиночных константных неисправностей $L(X)$ заключается в дизъюнкции списков транспортируемых неисправностей, записанных по единичным значениям координат строки таблицы производных, умноженной на отрицание конъюнкций, записанных по нулевым координатам строки таблицы производных:

$$L^S(X) = \bigvee_{i=1}^{2^n} \left\{ \left[\bigvee_{X_{ij}=1} L(X_{ij}) \right] \& \left[\overline{\bigwedge_{X_{ij}=0} L(X_{ij})} \right] \right\} =$$

$$= \bigvee_{i=1}^{2^n} \left\{ \left[\bigvee_{X_{ij}=1} L(X_{ij}) \right] \& \left[\bigvee_{X_{ij}=0} \bar{L}(X_{ij}) \right] \right\}.$$

Следуя данной формуле, синтезируем таблицы выражений для дедуктивного моделирования одиночных неисправностей с числом термов, равным размерности или длине кубитного покрытия (табл. 7).

В табл. 7 запись термов ДНФ по единичным значениям столбцов-производных (правая часть) представляет собой приближенные условия для транспортирования списков входных одиночных неисправностей на

Таблица 5

1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	0	1	0
0	0	1	0	1	1	1	1	1	1	0	0
0	1	0	0	1	1	1	0	1	1	1	0
1	0	0	0	1	1	1	1	0	1	1	0
1	1	1	1	0	0	0	1	1	1	1	1

Таблица 6

1	2	3	4	5	6	7	8	9	10	11	12	FD	FC
1	1	1	1	.	.	.	0	0	0	0	0	37	37
.	.	.	0	.	0	.	.	.	1	.	1	16	54
.	.	0	.	.	.	0	.	.	.	1	1	16	66
.	0	.	.	0	.	.	1	.	.	.	1	16	79
0	.	.	.	.	0	.	.	1	.	.	1	16	87
0	0	0	0	1	1	1	0	0	0	0	0	50	100

входных наборах, задаваемых соответствующими адресами строк (0000, 0001, …). Например, первая и последняя строки объединяют все списки входных неисправностей. Вторая строка свидетельствует о транспортировании на выход функциональности всех дефектов от первого входа за вычетом всех тех, которые будут одновременно присутствовать на входах 2, 3, 4. При наличии в строке всех нулевых координат выполняется логическое пересечение списков одиночных неисправностей для всех входов, которые будут транспортироваться на выход функциональности. Точные формулы дедуктивного моделирования имеют большую размерность, где для каждого входного вектора необходимо строить не один терм, а ДНФ, верхним пределом размера которой является таблица истинности.

На рис. 10 представлены списки-векторы входных дефектов, которые необходимо транспортировать через примитив, схему Шнейдера, на шести тестовых наборах: 0000, 0001, 0010, 0100, 1000, 1111. Один результат, вектор выходных неисправностей, проверяемых на входном наборе 0000,

Таблица 7

x_1	x_2	x_3	x_4	Y	X_1	X_2	X_3	X_4	$L^S(X)$
0	0	0	0	1	1	1	1	1	$X_1 \vee X_2 \vee X_3 \vee X_4$
0	0	0	1	0	0	0	0	1	$X_4(\overline{X_1X_2X_3})$
0	0	1	0	0	0	0	1	0	$X_3(\overline{X_1X_2X_4})$
0	0	1	1	0	0	0	0	0	$X_1X_2X_3X_4$
0	1	0	0	0	0	1	0	0	$X_2(\overline{X_1X_3X_4})$
0	1	0	1	0	0	0	0	0	$X_1X_2X_3X_4$
0	1	1	0	0	0	0	0	0	$X_1X_2X_3X_4$
0	1	0	1	0	1	0	0	0	$X_1(\overline{X_2X_3X_4})$
1	0	0	0	0	1	0	0	0	$X_1(\overline{X_2X_3X_4})$
1	0	0	1	0	0	0	0	0	$X_1X_2X_3X_4$
1	0	1	0	0	0	0	0	0	$X_1X_2X_3X_4$
1	0	1	1	0	0	1	0	0	$X_2(\overline{X_1X_3X_4})$
1	1	0	0	0	0	0	0	0	$X_1X_2X_3X_4$
1	1	0	1	0	0	0	1	0	$X_3(\overline{X_1X_2X_4})$
1	1	1	0	0	0	0	0	1	$X_4(\overline{X_1X_2X_3})$
1	1	1	1	0	1	1	1	1	$X_1 \vee X_2 \vee X_3 \vee X_4$

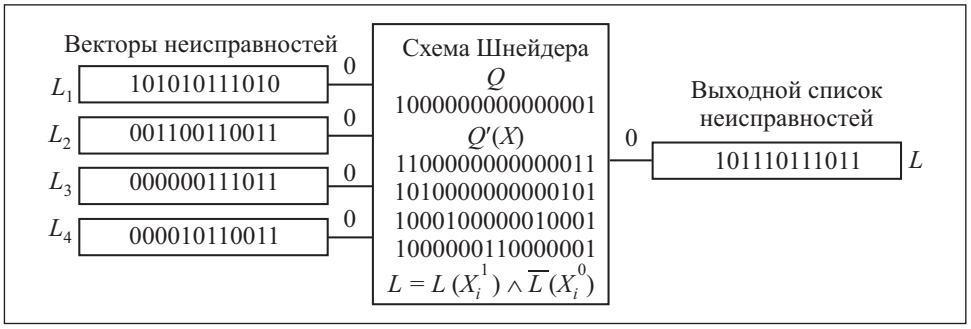


Рис. 10. Транспортирование векторов неисправностей

получен в результате использования первого столбца матрицы векторных производных $Q'(X)$ для моделирования одиночных константных дефектов. Естественно, что каждая неисправность из входного списка вектора является инверсной по отношению к тестовому состоянию рассматриваемой входной переменной.

Результаты моделирования входных неисправностей на шести входных наборах (Test Column) с использованием столбцов матрицы производных (Derivatives) для примера Шнейдера имеют следующий вид (A — входные неисправности; B — тест-столбцы; C — кубитные производные; D — выходной список неисправностей):

A												B						C					
1	0	1	0	1	0	1	1	1	0	1	0	0	1	0	0	0	1	1	1	0	0	0	1
0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	0	0	1	1	0	1	0	0	1
0	0	0	0	0	0	1	1	1	0	1	1	0	0	0	1	0	1	1	0	0	1	0	1
1	1	1	1	0	1	0	0	1	1	0	0	0	0	0	0	1	1	1	0	0	0	1	1

=

D

=

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

Недостатком технологии кубитно-дедуктивного моделирования по кубитным производным является незначительное уменьшение точности. Это влияет на теоретические результаты моделирования. Для сравнения предлагаются таблицы построения точных формул дедуктивного анализа для семи тестовых входных последовательностей схемы Шнейдера. Левый столбец табл. 8 представляет собой таблицу истинности. Под табл. 8 указаны входной тестовый вектор, а также две равнозначные формулы дедуктивного моделирования одиночных константных неисправностей. В формировании термов дедуктивного анализа принимают участие только те строки таблицы истинности, которые задают единичные значения функции *Y*. Если входной тестовый набор обращает функциональность в единицу, то все дедуктивные термы подлежат инверсии.

Сокращенные таблицы, из которых исключены строки, не существенные при получении результатов в виде аналитических выражений для

Таблица 8

X_1	X_2	X_3	X_4	Y	L_i	$L_i \oplus Y(x)$	ДНФ	$L^S(x, X)$
0	0	0	0	1	1111	$\overline{1111}$	$(\overline{X_1 X_2 X_3 X_4})$	$\overline{X}_1 \vee \overline{X}_2 \vee \overline{X}_3 \vee \overline{X}_4$
0	0	0	1	0	.	.	.	.
0	0	1	0	0	.	.	.	.
0	0	1	1	0	.	.	.	.
0	1	0	0	0	.	.	.	.
0	1	0	1	0	.	.	.	.
0	1	1	0	0	.	.	.	.
0	1	1	1	0	.	.	.	.
1	0	0	0	0	.	.	.	.
1	0	0	1	0	.	.	.	.
1	0	1	0	0	.	.	.	.
1	0	1	1	0	.	.	.	.
1	1	0	0	0	.	.	.	.
1	1	0	1	0	.	.	.	.
1	1	1	0	0	.	.	.	.
1	1	1	1	1	0000	$\overline{0000}$	$(\overline{\overline{X}_1 \overline{X}_2 \overline{X}_3 \overline{X}_4})$	$X_1 \vee X_2 \vee X_3 \vee X_4$

$x = 1111$; $L_i = x(1111) \oplus X_i^1$; $Y(x) = 1$; $L_i = L_i \oplus Y(x) = \overline{L_i}$;
$L = (X_1 \vee X_2 \vee X_3 \vee X_4)(\overline{X}_1 \vee \overline{X}_2 \vee \overline{X}_3 \vee \overline{X}_4)$; $L = (X_1 \vee X_2 \vee X_3 \vee X_4)(\overline{X_1 X_2 X_3 X_4})$.

дедуктивного моделирования одиночных неисправностей функциональных компонентов, имеют следующий вид:

X_1	X_2	X_3	X_4	Y	L_i	$L_i \oplus Y(x)$	ДНФ	$L^S(x, X)$
0	0	0	0	1	0000	$\overline{0000}$	$\overline{\overline{X}_1\overline{X}_2\overline{X}_3\overline{X}_4}$	$X_1 \vee X_2 \vee X_3 \vee X_4$
0	0	0	1	0	.	.	.	.
.	.	.	.	.	.	.	.	.
1	1	1	1	1	1111	$\overline{1111}$	$\overline{X_1X_2X_3X_4}$	$\overline{X}_1 \vee \overline{X}_2 \vee \overline{X}_3 \vee \overline{X}_4$

$x = 0000$; $L_i = x(0000) \oplus X_i^1$; $Y(x) = 1$; $L_i = L_i \oplus Y(x) = \overline{L_i}$;
$L = (X_1 \vee X_2 \vee X_3 \vee X_4)(\overline{X}_1 \vee \overline{X}_2 \vee \overline{X}_3 \vee \overline{X}_4)$; $L = (X_1 \vee X_2 \vee X_3 \vee X_4)(\overline{X_1X_2X_3X_4})$.

X_1	X_2	X_3	X_4	Y	L_i	$L_i \oplus Y(x)$	ДНФ	$L^S(x, X)$
0	0	0	0	1	0001	0001	$\overline{X}_1\overline{X}_2\overline{X}_3X_4$	$(\overline{X_1 \vee X_2 \vee X_3})X_4$
0	0	0	1	0	.	.	.	.
.	.	.	.	.	.	.	.	.
1	1	1	1	1	1110	1110	$X_1X_2X_3\overline{X}_4$	$(X_1X_2X_3)\overline{X}_4$

$x = 0001$; $L_i = x(0001) \oplus X_i^1$; $Y(x) = 0$; $L_i = L_i \oplus Y(x) = L_i$;
$L = \overline{X}_1\overline{X}_2\overline{X}_3X_4 \vee X_1X_2X_3\overline{X}_4$; $L = (\overline{X_1 \vee X_2 \vee X_3})X_4 \vee (X_1X_2X_3)\overline{X}_4$.

X_1	X_2	X_3	X_4	Y	L_i	$Y(x) = 0 \to L_i$	ДНФ	$L^S(x, X)$
0	0	0	0	1	1100	1100	$X_1X_2\overline{X}_3\overline{X}_4$	$X_1X_2(\overline{X_3 \vee X_4})$
0	0	0	1	0	.	.	.	.
.	.	.	.	.	.	.	.	.
1	1	1	1	1	0011	0011	$\overline{X}_1\overline{X}_2X_3X_4$	$(\overline{X_1 \vee X_2})X_3X_4$

$x = 1100$; $L_i = x(1100) \oplus X_i^1$; $L_i = L_i \oplus Y(x) = L_i$;
$L = X_1X_2\overline{X}_3\overline{X}_4 \vee \overline{X}_1\overline{X}_2X_3X_4$; $L = X_1X_2(\overline{X_3 \vee X_4}) \vee (\overline{X_1 \vee X_2})X_3X_4$.

X_1	X_2	X_3	X_4	Y	L_i	$L_i \oplus Y(x)$	ДНФ	$L^S(x, X)$
0	0	0	0	1	0010	0010	$\overline{X}_1\overline{X}_2X_3\overline{X}_4$	$(\overline{X_1 \vee X_2 \vee X_4})X_3$
0	0	0	1	0	.	.	.	.
.	.	.	.	.	.	.	.	.
1	1	1	1	1	1101	1101	$X_1X_2\overline{X}_3X_4$	$(X_1X_2X_4)\overline{X}_3$

$x = 0010$; $L_i = x(0010) \oplus X_i^1$; $Y(x) = 0$; $L_i = L_i \oplus Y(x) = L_i$;
$L = \overline{X}_1\overline{X}_2X_3\overline{X}_4 \vee X_1X_2\overline{X}_3X_4$; $L = (\overline{X_1 \vee X_2 \vee X_4})X_3 \vee (X_1X_2X_4)\overline{X}_3$.

X_1	X_2	X_3	X_4	Y	L_i	$L_i \oplus Y(x)$	ДНФ	$L^S(x, X)$
0	0	0	0	1	0100	0100	$\overline{X}_1 X_2 \overline{X}_3 \overline{X}_4$	$(\overline{X_1 \vee X_3 \vee X_4})X_2$
0	0	0	1	0	.	.	.	.
.	.	.	.	.	.	.	.	.
1	1	1	1	1	1011	1011	$X_1 \overline{X}_2 X_3 X_4$	$(X_1 X_3 X_4)\overline{X}_2$

$x = 0100;\ L_i = x(0100) \oplus X_i^1;\ Y(x) = 0;\ L_i = L_i \oplus Y(x) = L_i;$
$L = \overline{X}_1 X_2 \overline{X}_3 \overline{X}_4 \vee X_1 \overline{X}_2 X_3 X_4;\ L = (\overline{X_1 \vee X_3 \vee X_4})X_2 \vee (X_1 X_3 X_4)\overline{X}_2.$

X_1	X_2	X_3	X_4	Y	L_i	$L_i \oplus Y(x)$	ДНФ	$L^S(x, X)$
0	0	0	0	1	1000	1000	$X_1 \overline{X}_2 \overline{X}_3 \overline{X}_4$	$(\overline{X_2 \vee X_3 \vee X_4})X_1$
0	0	0	1	0	.	.	.	.
.	.	.	.	.	.	.	.	.
1	1	1	1	1	0111	0111	$\overline{X}_1 X_2 X_3 X_4$	$(X_2 X_3 X_4)\overline{X}_1$

$x = 1000;\ L_i = x(1000) \oplus X_i^1;\ Y(x) = 0;\ L_i = L_i \oplus Y(x) = L_i;$
$L = X_1 \overline{X}_2 \overline{X}_3 \overline{X}_4 \vee \overline{X}_1 X_2 X_3 X_4;\ L = (\overline{X_2 \vee X_3 \vee X_4})X_1 \vee (X_2 X_3 X_4)\overline{X}_1.$

Использование полученных точных формул дедуктивного моделирования четырех списков неисправностей на шести входных наборах для схемы Шнейдера дает такой же результат, как был получен с помощью анализа дефектов на основе Q-векторов булевых производных, а именно:

IF	DF	OF
101010111010	$L = 0000 \wedge (X_1 \vee X_2 \vee X_3 \vee X_4)(\overline{X_1 X_2 X_3 X_4})$	111111111111
001100110011	$L = 0001 \wedge (\overline{X_1 \vee X_2 \vee X_3})X_4 \vee (X_1 X_2 X_3)\overline{X}_4$	000010000000
000000111011	$L = 0010 \wedge (\overline{X_1 \vee X_2 \vee X_4})X_3 \vee (X_1 X_2 X_4)\overline{X}_3$	000000000000
111101001100	$L = 0100 \wedge (\overline{X_1 \vee X_3 \vee X_4})X_2 \vee (X_1 X_3 X_4)\overline{X}_2$	000000000000
	$L = 1000 \wedge (\overline{X_2 \vee X_3 \vee X_4})X_1 \vee (X_2 X_3 X_4)\overline{X}_1$	010001000100
	$L = 1111 \wedge (X_1 \vee X_2 \vee X_3 \vee X_4)(\overline{X_1 X_2 X_3 X_4})$	111111111111

Здесь IF — входные неисправности; DF — дедуктивные формулы; OF — выходные неисправности.

Таким образом, для повышения производительности дедуктивного анализа неисправностей можно пожертвовать несколькими процентами точности, которая не оказывает существенного влияния при проектировании цифровых систем большой размерности. Неточность кубитно-дедуктивного метода связана с выявлением неисправностей, которые являются

следствием многомерной активизации дефектов на линиях сходящихся разветвлений. Выигрышем от использования практически ориентированного метода является существенное уменьшение затрат памяти для хранения структур данных, а также технологичность и высокое быстродействие реализации алгоритма дедуктивного анализа, основанного только на использовании векторов кубитных производных.

Кубитная форма описания цифровых систем по метрике (компактность, быстродействие и качество) превосходит все существующие способы задания вычислительных устройств. Кубитное покрытие функциональности является наиболее технологичным средством для решения задач анализа, синтеза, тестирования и моделирования цифровых компонентов. Учитывая тот факт, что кубитные покрытия большой размерности, как правило, определены не по всем координатам, иногда целесообразно создавать компактный кубит-вектор посредством введения избыточности в виде вектора-дешифратора фактических и модельных адресов. Например, кубитное покрытие с неопределенными значениями (10xxxx0xxxxxxxx1) можно записать компактно по существенным состояниям разрядов как (1010), сохранив исходные адреса битов в дополнительном векторе (0, 1, 6, 15). Кроме того, основываясь на кубитной теории проектирования и тестирования, необходимо отказываться от моделей неисправностей для логических схем посредством создания новых методов моделирования кубитных дефектов.

Выводы

Метод взятия производных для синтеза тестов функциональных компонентов обеспечивает параллельное выполнение регистровых логических операций (shift, or, not, nxor) над кубитным вектором, что дает возможность существенно уменьшить время генерирования входных наборов и тестирования устройства посредством уменьшения аппаратной избыточности.

Дедуктивный метод моделирования неисправностей для функциональных компонентов обеспечивает параллельное выполнение регистровых логических операций (shift, or, not, nxor) над кубитным вектором и его производными, что дает возможность существенно уменьшить время верификации и тестирования цифрового устройства в режиме embedded online.

Процессор кубитного моделирования цифровых устройств, имплементированный в SoC или Cloud Service, для анализа исправного поведения и неисправностей на основе использования кубитных покрытий функциональных элементов, отличается от известных реализаций применением минимального набора параллельных регистровых логических операций и высоким быстродействием.

Предложенный метод синтеза тестов для функциональностей на основе кубитного покрытия может быть использован в качестве встроенного BIST-компонента для сервисного обслуживания SoC на основе стандарта граничного сканирования IEEE 1500 SECT или в качестве облачного online сервиса тестирования аппаратных модулей посредством IP протокола.

Направления будущих исследований: создание теории кубитного проектирования и тестирования цифровых систем на кристаллах; разработка облачных сервисов кубитного синтеза и анализа цифровых систем на кристаллах; создание программно-аппаратных генераторов тестов, симуляторов неисправностей, исправного поведения, алгоритмов диагностирования и библиотечных решений, встроенных в инфраструктуру кристаллов и (или) облачные сервисы, использующих кубитное описание функциональности логического компонента.

СПИСОК ЛИТЕРАТУРЫ

1. *Hahanov V., Litvinova E., Gharibi W. et al*. Quantum Memory-driven Computing for Test Synthesis// Proc. of IEEE East-West Design & Test Symposium (EWDTS'2017). Serbia, Novi Sad, 2017, p. 63-68. ISBN 978-1-5386-3298-7.
2. *Хаханов В.И., Ваджеб Гариби, Литвинова Е.И., Шкиль А.С.* Кубитные структуры данных вычислительных устройств // Электрон. моделирование. 2015, **37**, № 1, с. 76—99.
3. *Хаханов В.И., Тамер Бани Амер, Чумаченко С.В., Литвинова Е.И.* Кубитные технологии анализа и диагностирования цифровых устройств // Там же. 2015, **37**, № 3, с. 17—40.
4. *Skobtsov Yu.A., Speransky D.V.* Analytical method for synthesis recognizing sequences for discrete devices with memory // Autom. Remote Control, 41:1. 1980, p. 97—104.
5. *Хаханов В.И., Литвинова Е.И., Хаханова И.В., Гузь О.А.* Проектирование и тестирование цифровых систем на кристаллах. Харьков : ХНУРЭ, 2009, 484 с.
6. *Pomeranz I., Reddy S.M.* Aliasing Computation Using Fault Simulation with Fault Dropping // IEEE Transaction on Computers. 1995, p. 139—144.
7. *Ubar R., K͡ousaar J., Gorev M., Devadze S*. Combinational fault simulation in sequential circuits // 2015 IEEE International Symposium on Circuits and Systems (ISCAS). Lisbon, 2015, p. 2876—2879.
8. *Gorev M., Ubar R., 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.
9. *Pomeranz I.* Fault simulation with test switching for static test compaction // 2014 IEEE 32nd VLSI Test Symposium (VTS). Napa, CA, 2014, p. 1—6.
10. *Molnar L., Gontean A.* Fault simulation methodes // 2016 12th IEEE International Symposium on Electronics and Telecommunications (ISETC). Timisoara, Romania. 2016, p. 194—197.
11. *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.
12. *Mirkhani S., 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.

Поступила 03.05.17;
после доработки 12.09.17

REFERENCES

1. Hahanov, V., Litvinova, E., Gharibi, W. et al (2017), "Quantum memory-driven computing for test synthesis", *Proceedings of IEEE East-West Design & Test Symposium* (EWDTS'2017), Serbia, Novi Sad, 2017, pp. 63-68.
2. Hahanov, V., Gharibi, W., Litvinova, E. and Shkil, A.S. (2015), "Qubit structures of data structure of computing devices", *Elektronnoe modelirovanie,* Vol. 37, no. 1, pp. 76-99.
3. Hahanov, V., Tamer Bani Amer, Chumachenko, S.V. and Litvinova, E.I. (2015), "Qubit technologies of analysis and diagnosis of digital devices", *Elektronnoe modelirovanie,* Vol. 37, no. 3, pp. 17-40.
4. Skobtsov, Yu. A. and Speransky, D.V. (1980), "Analytical method for synthesis recognizing sequences for discrete devices with memory", *Autom. Remote Control*, Vol. 41, no. 1, pp. 97-104.
5. Hahanov, V.I., Litvinova, E.I., Hahanova, I.V. and Guz, O.A. (2009), *Proektirovanie i testirovanie tsifrovykh system na kristallakh* [Design and testing of digital systems on crystals], KhNURE, Kharkov, Ukraine.
6. Pomeranz, I. and Reddy, S.M. (1995), "Aliasing computation using fault simulation with fault dropping", *IEEE Transactions on Computers*, pp. 139-144.
7. Ubar, R., Kõusaar, J., Gorev, M. and Devadze, S. (2015), "Combinational fault simulation in sequential circuits", *IEEE International Symposium on Circuits and Systems* (ISCAS), 2015, Lisbon, pp. 2876-2879.
8. Gorev, M., Ubar, R. and Devadze, S. (2015), "Fault simulation with parallel exact critical path tracing in multiple core environment", *Proceedings of Design, Automation & Test in Europe Conference & Exhibition* (DATE), 2015, Grenoble, pp. 1180-1185.
9. Pomeranz, I. (2014), "Fault simulation with test switching for static test compaction", *Proceedings of IEEE 32nd VLSI Test Symposium* (VTS), 2014, Napa, CA, pp. 1-6.
10. Molnar, L. and Gontean, A. (2016), "Fault simulation methodes", *Proceedings of the 12th IEEE International Symposium on Electronics and Telecommunications* (ISETC), 2016, Timisoara, Romania, pp. 194-197.
11. Hadjitheophanous, S., Neophytou, S.N. and Michael, M.K. (2016), "Scalable parallel fault simulation for shared-memory multiprocessor systems", *Proceedings of the IEEE 34th VLSI Test Symposium* (VTS), 2016, Las Vegas, NV, pp. 1-6.
12. Mirkhani, S. and Abraham, J.A. (2014), "EAGLE: A regression model for fault coverage estimation using a simulation based metric", *Proceedings of the International Test Conference*, 2014, Seattle, WA, pp. 1-10.

Reseived 03.01.17;
affer revision 12.09.17

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

QUBIT METHOD FOR DEDUCTIVE FAULT ANALYSIS OF LOGIC CIRCUITS

Innovative methods have been developed for taking Boolean derivatives, test synthesis on their basis, as well as deductive fault simulation for functional elements specified by the qubit coverage. The analysis methods use vector logical operations: and, or, not, xor, as well as opposite shift of the parts of the qubit form of functionality. Examples of combinational circuits for verification and comparative analysis of the performance of basic and proposed methods are presented. The structure of the embedded processor is described, which executes the operations of taking derivatives, test synthesis, deductive fault simulation for evaluating the quality of input test patterns and

diagnosis. The proposed technologies are focused on their implementation in a cloud service or IP-infrastructure of SoC architectures.

K e y w o r d s: test synthesis, SoC design and verification, qubit coverage, digital circuit, fault simulation, Boolean derivative, deductive fault simulation.

ХАХАНОВ Владимир Иванович, д-р техн. наук, профессор, гл. науч. сотр., профессор кафедры автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники. В 1978 г. окончил Харьковский ин-т радиоэлектроники. Область научных исследований — компьютерная инженерия, киберфизические системы и облачный компьютинг.

ЕМЕЛЬЯНОВ Игорь Валерьевич, аспирант кафедры автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники, который окончил в 2007 г. Область научных исследований — компьютерные системы и сервис-компьютинг.

ЛЮБАРСКИЙ Михаил Михайлович, аспирант кафедры автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники. В 2010 г. окончил Харьковский национальный университет имени В.Н. Каразина. Область научных исследований — компьютерные системы и сервис-компьютинг.

ЧУМАЧЕНКО Светлана Викторовна, д-р техн. наук, профессор, зав. кафедрой автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники. В 1991 г. окончила Харьковский национальный университет имени В.Н. Каразина. Область научных исследований — математическое моделирование вычислительных процессов.

ЛИТВИНОВА Евгения Ивановна, д-р техн. наук, профессор, 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.

ТАМЕР БАНИ АМЕР, аспирант кафедры автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники. В 2014 г. окончил Донецкий национальный технический университет. Область научных исследований — компьютерные системы и сервис-компьютинг.