УДК681.326:519.713

СИНТЕЗ Q-ТЕСТОВ ПО КУБИТНОМУ ОПИСАНИЮ ФУНКЦИОНАЛЬНОСТЕЙ

TAMER BANI AMER, ЕМЕЛЬЯНОВ И.В., ЛЮБАРСКИЙ М., ХАХАНОВ В.И.

Предлагаются методы и аппаратно-программные реализации безусловного параллельного синтеза тестов на основе булевых производных для логических схем в black box форме, заданных кубитным покрытием. Дается теоретическое обоснование применения методов и оценки их эффективностей для широкого класса цифровых схем, имплементируемых в кристаллы программируемых логических устройств. Предлагаются инновационные методы взятия булевых производных и дедуктивного моделирования неисправностей для функциональных элементов, заданных кубитными покрытиями.

1. Реализация логических функциональностей на элементах памяти

Понятие адресного выполнения логических операций, реализованных на элементах памяти LUT в программируемых логических устройствах (PLD), дает потенциальную возможность создавать на кристалле только адресное пространство, максимально технологичное для встроенного восстановления работоспособности всех компонентов, участвующих в формировании функциональности [1-3]. Актуальность создания адресного пространства для всех компонентов подтверждается следующим распределением логики и памяти на кристалле, представленным на рис. 1, где после 2020 года на чипе будет только один процент логики и 99 процентов памяти.

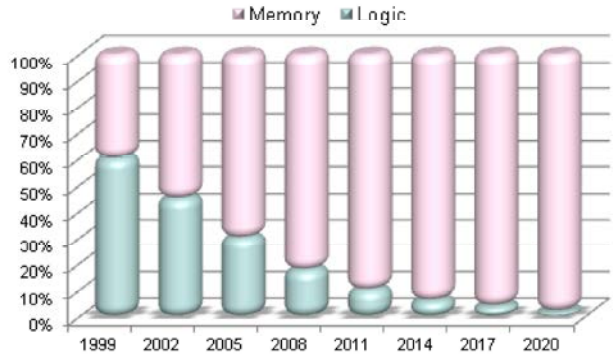

Рис. 1. Распределение памяти и логики на кристалле

Тенденция к увеличению памяти дает возможность встроенного восстановления работоспособности отказавших ячеек за счет выделенных дополнительных ресурсов для их ремонта (spare logic cells). Проблема автономного устранения дефектов (самовосстановления работоспособности) логических элементов связана с отсутствием у них адресов. Но решить ее можно, если связи между элементами логики сделать гибкими с помощью программы описания структуры, помещенной в память, которая соединит логические компоненты в схему. Кроме структуры взаимодействия элементов память должна содержать порядок их обработки. В случае возникновения дефекта в одном из адресуемых логических элементов система встроенного тестирования восстановит его работоспособность путем переадресации на заведомо исправный аналог из ремонтного запаса. Просто решается проблема повышения качества и надежности цифровых систем на кристаллах путем создания инфраструктуры встроенного тестирования, диагностирования, оптимизации и восстановления работоспособности за счет аппаратной избыточности и уменьшения быстродействия выполнения функциональных операций [2-5].

Цель – существенное повышение быстродействия синтеза тестов и дедуктивной верификации для black box функциональностей логических компонентов с помощью компактных описаний в форме кубитных покрытий и параллельного выполнения минимального числа регистровых логических операций (shift, or, not, nxor).

Задачи:

1) Метод генерации тестов для black box функциональностей логических схем на основе использования кубитных покрытий и параллельного выполнения регистровых логических операций (shift, or, not, nxor).

2) Метод взятия булевых производных для синтеза тестов на основе использования кубитных покрытий.

3) Метод синтеза тестов на основе применения булевых производных, представленных векторами в формате кубитных покрытий.

Сущность исследования – разработка методов генерирования входных тестовых последовательностей и оценки их качества для функциональных логических компонентов путем параллельного выполнения регистровых логических операций (shift, or, not, nxor) над кубитным покрытием и его производными в структуре процессора кубитного моделирования.

Мотивация нового подхода для проектирования компьютинговых систем обусловлена появлением облачных сервисов в рамках новой киберкультуры Internet of Things, которая представляет собой специализированные и рассредоточенные в пространстве системы, реализуемые в аппаратуре или в программном продукте. Любой компонент функциональности, равно как и структура системы, представляется векторной формой, упорядоченной по адресам, таблицы истинности, реализуемой с помощью памяти. Логические функции в традиционном исполнении reusable logic не рассматриваются. От этого частично уменьшается быстродействие, но, учитывая, что 94% SoC-кристалла составляет память [2,3], оставшиеся 6% будут имплементироваться в памяти, что не критично для большинства облачных сервисов. Практически для создания эффективных компьютинговых структур следует использовать теорию, основанную на вычислительных компонентах высокого уровня абстракции: адресуемая память и транзакция.

Особенность организации данных в классическом компьютере состоит в адресации бита, байта или друго-

го компонента. Адресуемость создает проблему в обработке ассоциации неадресуемых элементов множества, которые не имеют порядка по определению. Решением может быть процессор, где образ универсума из n унитарно кодированных примитивов использует суперпозицию для формирования булеана $|B(A)| = 2^n$ всех возможных состояний [4,12].

Следовательно, структуру данных «булеан» можно рассматривать как детерминированный образ квантового кубита в алгебре логики, элементы которой унитарно кодируются двоичными векторами и обладают свойствами суперпозиции, параллелизма и перепутывания. Это дает возможность использовать предлагаемые кубитные модели для повышения быстродействия анализа цифровых устройств на классических вычислителях, а также без модификации – в квантовых компьютерах, которые появятся через несколько лет на рынке электроники.

Квантовое описание цифровых функциональных элементов. Кубит (n-кубит) есть векторная форма унитарного кодирования универсума из n примитивов для задания булеана состояний 2^{2^n} с помощью 2^n двоичных переменных. Если n=2, то 2-кубит задает 16 состояний с помощью четырех переменных, при n=1 кубит задает четыре состояния на универсуме из двух примитивов (10) и (01) с помощью двух двоичных переменных (00,01,10,11) [12]. При этом допускается суперпозиция в векторе 2^n состояний, обозначенных примитивами. Синонимом кубита при задании двоичного вектора логической функции является Q-покрытие (Q-вектор) [6,7] как унифицированная векторная форма суперпозиционного задания выходных состояний, соответствующих адресным кодам входных переменных функционального элемента. Формат структурного кубитного компонента цифровой схемы $Q^* = (X, Q, Y)$ включает интерфейс (входные и выходную переменные), а также кубит-вектор Q, задающий функцию $Y = Q(X)$, размерность которого определяется степенной функцией от числа входных линий $k = 2^n$. Новизна кубитной формы заключается в замене неупорядоченных по строкам таблиц истинности функциональных элементов векторами упорядоченных состояний выходов. Например, если функциональный примитив имеет двоичную таблицу, то ему можно поставить в соответствие кубит или Q-покрытие: Q=(1110):

$$C = \begin{array}{ccc} X_1 & X_2 & Y \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{array} \rightarrow Q = (1110).$$

Таким образом, дизрапторная идея, положенная в основу исследований, заключается в замене множества вход-выходных соответствий таблицы истинности кубитным вектором адресуемых выходных состояний (рис. 2).

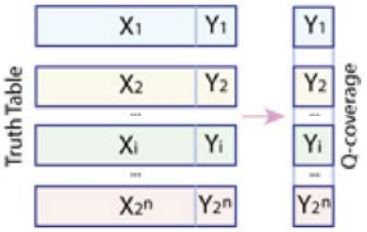

Рис. 2. Таблица истинности и кубитное покрытие

Примитивизм и компактность кубитной векторной формы (Q-coverage) диктует применение только простых параллельных регистровых операций над его содержимым: (not, shift, or, and, xor) для решения всех задач синтеза и анализа цифровых изделий.

2. Кубитный метод синтеза тестов

Предлагается метод синтеза тестов, использующий кубитные векторы или Q-покрытия функциональных примитивов цифровых устройств, который характеризуется компактностью описания данных и параллелизмом выполнения логических операций.

Q-покрытие есть векторная форма описания поведения цифрового устройства, где каждый разряд имеет адрес, формируемый двоичными состояниями его входных переменных:

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

Q-тест есть векторная форма неявного задания тестовых последовательностей цифрового устройства, где координаты вектора формируют упорядоченную последовательность двоичных наборов, подаваемых на входные переменные по правилу:

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

Другими словами, если координата вектора Q(i)=1, то тестовый набор, составленный из двоичных разрядов, формирующих десятичный адрес i, подается на входы устройства. В противном случае, при нулевом значении координаты Q(i)=0, такой тестовый набор отсутствует.

Естественно, что размерности Q-покрытия и Q-теста всегда одинаковы, что делает возможным выполнять параллельный анализ их совместного взаимодействия при синтезе тестов для функциональных элементов по правилу [4]: F⊕Q⊕T=0, T=Q⊕F.

Пример 1. Необходимо сгенерировать тест проверки одиночных константных неисправностей для логического элемента 2and, заданного таблицей истинности. Для этого задается таблица влияния одиночных константных неисправностей входных переменных с числом кубов, равным количеству входов. Каждый куб неисправности имеет единицу на координате входной переменной и на выходе, а остальные координаты равны нулю. Это означает, что существует заказ: изменение входной переменной должно вызывать изменение состояния выхода.

Алгоритм синтеза теста по таблице истинности [4]:

1) На первом шаге выполняется покоординатная xor-операция между таблицей истинности и каждой строкой матрицы неисправностей, которая генерирует две таблицы, по числу входных переменных, содержащие строки-кандидаты в тест:

$$\begin{bmatrix} X_1 & X_2 & Y \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \oplus \begin{bmatrix} F_1 & F_2 & F_Y \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} X_1 & X_2 & \overline{Y}_1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \vee \begin{bmatrix} X_1 & X_2 & \overline{Y}_2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

2) Затем строки двух последних таблиц упорядочиваются по правилу возрастания двоичных кодов входных переменных:

$$\begin{bmatrix} X_1 & X_2 & \overline{Y}_1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \vee \begin{bmatrix} X_1 & X_2 & \overline{Y}_2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} =$$

$$= \begin{bmatrix} X_1 & X_2 & Y'_1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} X_1 & X_2 & Y'_2 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

3) После этого сравниваются полученные состояния выходов $(Y'_1Y'_2)$ с вектором значений выходов исходной таблицы истинности Y по правилу операции эквивалентности (nxor) – если значения одноименных координат двух кубитов равны, то результат сравнения равен 1, в противном случае – 0:

$$\begin{bmatrix} Y \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \overline{\oplus} \begin{bmatrix} Y'_1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \vee \begin{bmatrix} Y'_2 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} Y^t_1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} Y^t_2 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

4) На последнем шаге выполняется операция логического сложения полученных результатов сравнения, которая дает Q-тест или T-вектор, единичные координаты которого идентифицируют только те двоичные входные последовательности, которые следует подавать на входы цифрового устройства для его проверки:

$$\begin{bmatrix} Y^t_1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \vee \begin{bmatrix} Y^t_2 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} T \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} X_1 & X_2 & Y \\ . & . & . \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

В данном примере Q-тест содержит три единицы 0111, поэтому соответствующие ему входные наборы представлены тремя векторами: 010, 100, 111.

Пример 2 иллюстрирует работу алгоритма [4] синтеза теста по таблице истинности для функционального элемента, имеющего три входных переменных. Здесь также фигурирует таблица истинности, в которой последний столбец Y является кубит-вектором. Столбцы $(F_1F_2F_3F_Y)$ формируют матрицу неисправностей, сущность каждой строки которой – одномерная активизация входа на выход. Столбцы $(Y'_1Y'_2Y'_3)$ получены в результате xor-взаимодействия таблицы истинности и матрицы неисправностей, которое определяет поведение функциональности при внесении на каждый вход неисправности, инверсной по отношению к ее исправному поведению. Столбцы $(Y^t_1Y^t_2Y^t_3)$ получены путем покоординатного сравнения вектора выходных состояний исходной таблицы истинности с наборами $(Y'_1Y'_2Y'_3)$, полученными на предыдущем шаге. Вектор T – логически объединяет наборы $(Y^t_1Y^t_2Y^t_3)$ в тест для заданной функциональности:

X_1	X_2	X_3	Y	F_1	F_2	F_3	F_Y	Y'_1	Y'_2	Y'_3	Y^t_1	Y^t_2	Y^t_3	T
0	0	0	1	1	0	0	1	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	0	0	0	0	0
0	1	0	0	0	1	1	1	0	0	0	1	1	1	1
0	1	1	1					1	0	1	1	0	1	1
1	0	0	0					0	0	0	1	1	1	1
1	0	1	1					0	1	1	0	1	1	1
1	1	0	1					1	1	1	1	1	1	1
1	1	1	0					0	0	0	1	1	1	1

Последний столбец представляет собой тест проверки неисправностей внешних входных и выходных переменных функционального элемента T=(10111111). Тест-вектор задает следующие входные наборы, которые следует подать на входы функционального элемента: 0001, 0100, 0111, 1000, 1011, 1101, 1110, чтобы проверить все дефекты заданного класса.

Формальный Q-алгоритм синтеза тестов для константных неисправностей функционального примитива на основе использования Q-покрытия содержит следующие пункты:

1) Инвертирование Q-покрытия: $Q_i = \overline{Q}_i, i = \overline{1, 2^n}$.

2) Упорядочение инверсного Q-покрытия для каждой из n входных переменных $Q_j = S_j(\overline{Q}), j = \overline{1, n}$. Данная процедура сводится к выполнению операций логического сдвига (shift) на кубитном векторе, каждая из которых имеет собственный алгоритм для рассматриваемой входной переменной, что иллюстрируется следующей схемой для трех входов (рис. 3).

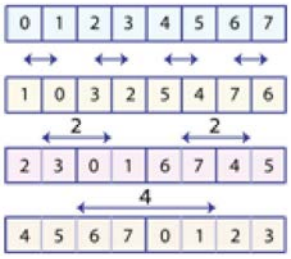

Рис. 3. Операции встречного сдвига на регистрах

Здесь вторая строка адресными индексами задает процедуру обмена данными между соседними координатами кубит-вектора путем встречного сдвига содержимого двух соседних координат, что формирует тест-вектор для первой входной переменной. Третья строка иллюстрирует обмен данными между соседними парами координат кубит-вектора путем встречного сдвига их содержимого, что формирует тест-вектор для второй входной переменной. Четвертая строка задает обмен данными между соседними тетрадами координат кубит-вектора путем встречного сдвига их содержимого, что формирует тест-вектор для третьей входной переменной.

3) Получение T-векторов для каждой из n входных переменных путем сравнения с исходным кубитным покрытием функционального элемента:

$$T_j = Q \oplus \overline{Q}_j, j = \overline{1,n}.$$

4) Получение Q-теста функциональности путем логического объединения T-векторов для каждой входной переменной:

$$T = \vee_{j=1}^{n} T_j.$$

Интегрально алгоритм безусловного синтеза тестов для функциональных элементов, заданных кубитными покрытиями, компактно может быть записан в виде следующей формулы:

$$T = \vee_{j=1}^{n} [Q \overline{\oplus}_{j=1}^{n} S_j(\overline{Q})].$$

Вычислительная сложность безусловного алгоритма синтеза тестов на основе последовательного использования регистровых логических операций (not – shift – nxor – or) для функционального элемента, описанного кубит-вектором, представлена следующим выражением:

$$q = n + n \times 2^n + n + n = n(2^n + 3).$$

Здесь n – число переменных, первое слагаемое определяет вычислительную сложность операции 1) инверсии, второе – 2) логический сдвиг для перестановки состояний координат кубит-вектора относительно каждой из n входных переменных, третье – 3) nxor-сравнение полученных тест-векторов с исходным кубит-покрытием функционального элемента, четвертое – 4) объединение тест-векторов для входных переменных.

Абстрагируясь от понятия таблицы истинности, далее предлагаем формальный безусловный алгоритм кубитного синтеза тестов для функциональных примитивов на основе Q-покрытия применительно к ранее рассмотренному примеру:

1) Инвертирование всех разрядов Q-покрытия функционального элемента, имеющего три входных переменных:

Q	1	0	0	1	0	0	1	0
$\overline{Q}$	0	1	1	0	1	1	0	1

2) Логический сдвиг номеров разрядов инвертированного кубитного вектора в соответствии с последовательностями номеров:

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

Здесь действует простое логическое правило: для первой входной (младшей) переменной выполняется параллельный обмен данными между соседними координатами, для второй входной переменной реализуется обмен данными, но уже между соседними парами координат, для третьей переменной выполняется обмен данными между соседними тетрадами координат вектора адресов.

Обобщение операций сдвига для функциональности, содержащей 4 переменных, представлено на рис. 4. Увеличение количества переменных принципиально не изменяет сущностей сдвига данных: встречный сдвиг двух битов, пар битов, тетрад битов, восьмерок и т.д.

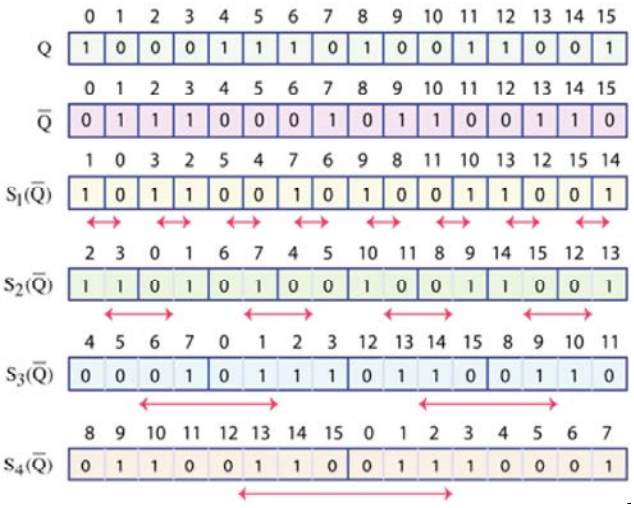

Рис. 4. Операции сдвига для четырех-входовой функциональности

Процедура встречного сдвига данных в инверсном кубитном векторе является самой времязатратной в алгоритме синтеза тестов для функциональных элементов. Поэтому ее быстродействие будет иметь максимальное значение при аппаратной реализации сдвиговых операций.

Процедурная реализация обмена номерами для формирования тест-векторов проверки неисправностей входных переменных, применительно к рассматриваемому примеру, имеет следующий вид:

Q	0	1	2	3	4	5	6	7
$\overline{Q}_1$	1	0	3	2	5	4	7	6
$\overline{Q}_2$	2	3	0	1	6	7	4	5
$\overline{Q}_3$	4	5	6	7	0	1	2	3

Интересно, что для первой переменной существует простая формула перенумерации ячеек кубит-вектора для синтеза тестов: $j = j + (-1)^j$. Для остальных переменных такой простой зависимости определить пока не удалось.

С учетом введенных правил встречного сдвига ячеек кубит-вектора реализация данного пункта алгоритма для Q=11010110 представлена таблицей:

Q =	1	1	0	1	0	1	1	0
$\overline{Q}$ =	0	0	1	0	1	0	0	1
$\overline{Q}_1$ =	1	0	0	1	0	0	1	0
$\overline{Q}_2$ =	1	1	0	1	0	1	1	0
$\overline{Q}_3$ =	0	0	0	1	0	1	1	0

Таким образом, каждая переменная делает разбиение кубит-вектора на группы упорядоченных последовательностей. Первая (младшая) переменная создает изменения состояний по правилу: прибавить 1 к текущему четному адресу ячейки: j=j+1, отнять 1 у текущего нечетного адреса ячейки: j=j-1. Для второй переменной рассматриваются уже пары ячеек, которые обрабатываются по правилу: прибавить 2 к текущему четному адресу пары ячеек: j=j+2, отнять 2 у текущего нечетного адреса пары ячеек: j=j-2. Для третьей переменной рассматриваются уже тетрады ячеек, которые обрабатываются по правилу: прибавить 4 к текущему четному адресу тетрады ячеек: j=j+4, отнять 4 у текущего нечетного адреса тетрады ячеек: j=j-4.

3) Сравнение с помощью операции эквивалентности полученных в данном случае трех инвертированных и переупорядоченных Q-векторов с исходным Q-покрытием функционального элемента:

Q	1	1	0	1	0	1	1	0
$\overline{Q}_1$	0	0	1	0	1	0	0	1
$S_1(\overline{Q})$	1	0	0	1	0	0	1	0
$S_2(\overline{Q})$	1	0	0	0	0	1	1	0
$S_3(\overline{Q})$	0	0	0	1	0	1	1	0
$Q \oplus S_1(\overline{Q})$	1	0	1	1	1	0	1	1
$Q \oplus S_2(\overline{Q})$	1	0	1	0	1	1	1	1
$Q \oplus S_3(\overline{Q})$	0	0	1	1	1	1	1	1

4) Дизъюнкция полученных трех векторов формирует Q-тест, единичные координаты которых определяют тестовые наборы для проверки всех одиночных константных неисправностей внешних входов и выходов:

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

Учитывая существенность встречных регистровых сдвигов, ниже предлагается универсальный алгоритм получения перестановок кубитных фрагментов в зависимости от номера входной переменной.

Последовательный алгоритм встречного сдвига данных в кубитных покрытиях для получения Q-тестов входных переменных имеет три вложенных цикла:

1) Задание i-номера входной переменной или шага 2^i для встречного сдвига данных в кубите: $i = \overline{1,n}$.

2) Формирование цикла обработки кубита с уже заданным шагом 2^i (2,4,8,16…), кратным степени двойки: $j = \overline{0, 2^n - 1}, 2^i$. В зависимости от номера входной переменной i формируются следующе последовательности индекса j=f(i): [(0,2,4,6…), (0,4,8,12…), (0,8,16,32…)].

3) Задание цикла встречного сдвига данных для пары соседних групп: $t = j, j + 2^{i-1} - 1$. Значения индекса t=f(i,j): обработка кубита первой переменной – (0,0; 2,2; 4,4…), второй переменной – (0,1; 4,5; 8,9…), третьей переменной – (0,3; 8,11; 16,19…). Выполнение операций сдвига посредством использования буферного регистра B (рис. 5):

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

Конец алгоритма сдвига данных в регистрах.

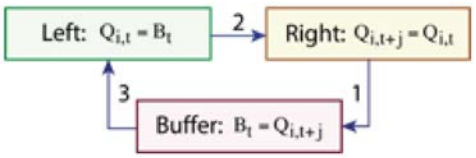

Рис. 5. Встречный сдвиг соседних частей кубита

Вычислительная сложность последовательного алгоритма для обработки кубитного покрытия функциональности, имеющей n входных переменных, равна $q = 3n2^n$.

Пример 3. Построить тест для проверки функциональности, заданной логическим уравнением: $Y = \overline{X}_1\overline{X}_2 \vee \overline{X}_1X_2$. Результаты выполнения алгоритма синтеза теста по заданному кубит-вектору представлены в следующем виде:

Q	0	0	1	1
$\overline{Q}_1$	1	1	0	0
$S_1(\overline{Q})$	0	0	1	1
$S_2(\overline{Q})$	1	1	0	0
$T_1 = Q \oplus S_1(\overline{Q})$	1	1	1	1
$T_2 = Q \oplus S_2(\overline{Q})$	0	0	0	0
$T = T_1 \vee T_2$	1	1	1	1

Результирующий тест содержит 4 входных последовательности, каждая из которых проверяет неисправности для входной переменной X1. Однако тест для переменной X_2 не существует, поскольку T2=0000. Это означает, что переменная X_2 не может быть проверена, а значит она не является существенной. Действительно, преобразование исходной функции

$$Y = \overline{X}_1\overline{X}_2 \vee \overline{X}_1X_2 = \overline{X}_1(\overline{X}_2 \vee X_2) = \overline{X}_1$$

в сторону минимизации исключает переменную X_2, как избыточную или несущественную. Таким образом, метод генерации тестов на основе кубитного покрытия дает дополнительную возможность опреде-

лять существенность переменных и минимизировать (уменьшать размерность) Q-вектор путем уменьшения количества переменных.

Пример 4. Построить тест для проверки функциональности, заданной логическим уравнением: $Y = \overline{X}_1 X_2 \vee X_1 \overline{X}_2$.Результаты выполнения алгоритма синтеза теста по заданному кубит-вектору на основе последовательного выполнения четырех регистровых логических операций (not, shift, nxor, or) представлены в виде:

Q	0 1 1 0
$\overline{Q}_1$	1 0 0 1
$S_1(\overline{Q})$	0 1 1 0
$S_2(\overline{Q})$	0 1 1 0
$T_1 = Q \overline{\oplus} S_1(\overline{Q})$	1 1 1 1
$T_2 = Q \overline{\oplus} S_2(\overline{Q})$	1 1 1 1
$T = T_1 \vee T_2$	1 1 1 1

Полученный тест содержит 4 входных последовательности, каждая из которых проверяет неисправности для входных переменных X_1, X_2. Таким образом, метод генерации тестов на основе кубитного покрытия позволяет сгенерировать тест для функционального компонента, но не предоставляет инструмента сделать его минимальным.

Пример 5. Синтезировать тест для проверки функциональности, заданной логическим уравнением: $Y = \overline{X}_1 X_2 \overline{X}_3 \vee X_1 \overline{X}_2 \overline{X}_3 \vee X_1 X_2 X_3 \vee \overline{X}_1 X_2 X_3$. Результаты выполнения алгоритма синтеза теста по Q-вектору на основе последовательного выполнения четырех регистровых логических операций (not, shift, nxor, or) представлены в виде:

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

На рис. 6 показан секвенсор синтеза тестов для функциональных элементов, заданных кубитными покрытиями. Он содержит модуль управления, который распределяет четыре синхроимпульса, создавая цикл генерации теста, включающий 4 параллельные операции, последовательно выполняемые: 0) Начальная стартовая операция, инициируемая сигналом Start, предполагает загрузку в регистр кубитного покрытия. 1) Затем синхроимпульс Clk N активирует выполнение операции инверсии Not над содержимым регистра кубитного покрытия. 2) Синхросигнал Clk S активирует выполнение операций встречного сдвига над содержимым, в данном случае трех, регистров, получая следующие результаты: S_1 (not Q), S_2 (not Q), S_3 (not Q). 3) Затем синхросигнал Clk C инициирует выполнение параллельных операций сравнения (в данном случае для трех регистров) полученных кандидатов в тест с начальным кубитным покрытием: nxor (notQ, S_1), nxor (notQ, S_2), nxor (notQ, S_3). 4) Синхросигнал Clk U активирует выполнение or-операции над кандидатами в тест, в данном случае реализацию логического объединения содержимого трех регистров:

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

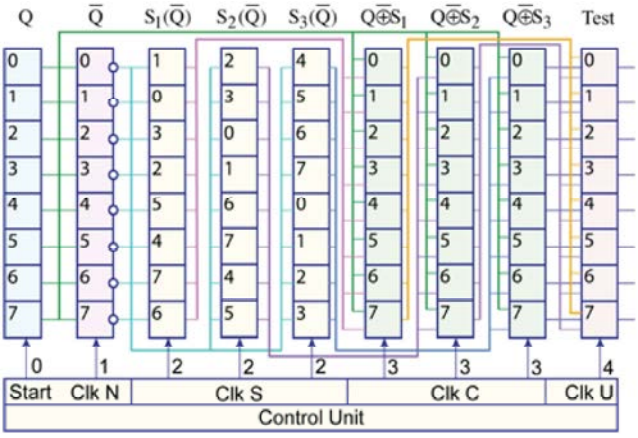

Рис. 6. Секвенсор синтеза тестов

Если не экономить на аппаратуре, то быстродействие тестового генератора можно довести до четырех автоматных тактов q=4 параллельного выполнения логических регистровых операций, что дает возможность встраивать секвенсор в BIST-инфраструктуру цифровых систем на кристаллах для online тестирования функциональных элементов. При этом суммарное количество 2^n-разрядных регистров будет равно N(n)=1+1+n+n+1=(2n+3), а общий объем регистровой памяти в битах будет равен N(R)=(2n+3) 2^n, где n – число входных переменных функционального элемента.

Сложить навстречу сдвинутые соседние тетрады кубита для вычисления производной по первой переменной. Сложить навстречу сдвинутые соседние пары кубита для вычисления производной по второй переменной. Сложить навстречу сдвинутые соседние биты кубита для вычисления производной по третьей переменной.

3. Вычисление булевых производных для Q-синтеза тестов

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 и векторной. Исследование метода выполним на примерах логических функций:

1) $f(x) = x_1 \vee x_1 \overline{x}_2$.

2) $f(x) = x_1 x_2 \vee \overline{x}_1 x_3$.

3) $f(x) = \overline{x}_2 \overline{x}_3 \vee x_1 x_2 x_3$.

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 6. Определить все производные первого порядка по кубитной форме логической функции $f(x) = x_1 \vee x_1\bar{x}_2$.

Применение формулы вычисления по аналитическому выражению

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

определяет булеву производную первого порядка как сумму по модулю два нулевой и единичной остаточных функций.

Для рассматриваемой функции получается:

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

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

Нулевая величина производной означает отсутствие условий активизации для переменной x_2, что дает основания считать ее несущественной, 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

Производная по таблице истинности может вычисляться путем 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 1, а по второй равна 0.

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

Пример 8. Определить все производные первого порядка по аналитической форме логической функции $f(x) = x_1x_2 \vee \bar{x}_1x_3$. Для рассматриваемой функции выполняются следующие вычисления:

$$\frac{df(x_1,x_2,x_3)}{dx_1} = f(0,x_2,x_3) \oplus f(1,x_2,x_3) =$$
$$= (0 \cdot x_2 \cdot \bar{0} \cdot x_3) \oplus (1 \cdot x_2 \cdot \bar{1} \cdot x_3) =$$
$$= (0 \cdot 1 \cdot x_3) \oplus (x_2 \cdot 0 \cdot x_3) =$$
$$= x_3 \oplus x_2 \oplus x_2\bar{x}_3 \oplus \bar{x}_2x_3.$$

$$\frac{df(x_1,x_2,x_3)}{dx_2} = f(x_1,0,x_3) \oplus f(x_1,1,x_3) =$$
$$= \bar{x}_1x_3 \oplus (x_1 \cdot x_3) \oplus$$
$$\overline{\bar{x}_1x_3}(x_1 \cdot x_3) \oplus \bar{x}_1x_3\overline{(x_1 \cdot x_3)} =$$
$$= (x_1 \cdot \bar{x}_3)(x_1 \cdot x_3) \oplus \bar{x}_1x_3\bar{x}_1\bar{x}_3 \oplus x_1.$$

$$\frac{df(x_1,x_2,x_3)}{dx_3} = f(x_1,x_2,0) \oplus f(x_1,x_2,1) =$$
$$= x_1x_2 \oplus (\bar{x}_1 \cdot x_2) \oplus$$
$$\overline{x_1x_2}(\bar{x}_1 \cdot x_2) \oplus x_1x_2\overline{(\bar{x}_1 \cdot x_2)} =$$
$$= (\bar{x}_1 \cdot \bar{x}_2)(\bar{x}_1 \cdot x_2) \oplus x_1x_2x_1\bar{x}_2 \oplus \bar{x}_1.$$

Для трех переменных получены 4 условия активизации, которые соответствуют 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_2'
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

Кубитный метод взятия булевой производной:

1) Определить кубитный вектор функциональности.

2) Выполнить xor-операцию для сдвинутых навстречу друг другу соседних частей кубита: биты, пары, тетрады.

3) Результат записать в обе соседние части: биты, пары, тетрады.

Для рассматриваемого примера выполнение процедуры взятия производной по первой переменной имеет вид: {a,b}=a⊕b, (0110,0110)=0101⊕0011. Для получения производной по второй переменной нужно последовательно xor-сложить соседние пары кубит-вектора Y, а общий результат записать в каждую пару: {a,b}=a⊕b: (00,00)=01⊕01, (11,11)=00⊕11. Для получения производной по третьей переменной необходимо последовательно xor-сложить соседние биты кубит-вектора Y, а общий результат записать в каждый бит соседей: {a,b}=a⊕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 8. 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 [4] приведенной функциональности на основе выполнения операций над столбцами входных переменных представлен в виде:

$\frac{df}{dx_1}$.

x_1	x_2	x_3	Y	Y_2^0	Y_2^1	Y_2'
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'
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

. $\bar{x}_1\bar{x}_3$. $x_1\bar{x}_3$. x_1x_3 . $\bar{x}_3$. x_1x_3.

$\frac{df}{dx_3}$.

x_1	x_2	x_3	Y	Y_3^0	Y_3^1	Y_3'
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

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

Как альтернатива, ниже приведены результаты выполнения процедур взятия производных первого порядка для трех переменных по кубическому покрытию (таблице истинности) 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

Далее представлен исключительно простой процесс вычисления производных на кубитном покрытии без рассмотрения таблицы истинности, что 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 (ДНФ) по единичным значениям переменных, формирующих адреса ячеек кубит-вектора:

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

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

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

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

Для сравнения далее коротко описан метод получения теста $T = [T_{ij}], i = \overline{1,k}; j = \overline{1,n}$ комбинационной функциональности, заданной кубическим покрытием или таблицей истинности, который содержит пункты [4]:

1) $f'(x_i) = f(x_1, x_2,...,x_i = 0,...,x_n) \oplus f(x_1, x_{2,...,}x_i = 1,...,x_n)$;

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

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

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

1) Вычисление производных по всем n переменным функциональности путем использования кубического покрытия. 2) Объединение всех условий (векторов) активизации в таблицу, где каждому вектору путем конкатенации (*) ставится в соответствие изменение переменной, по которой была взята производная, что означает удвоение числа тестовых наборов по отношению к общему количеству (k) условий активизации. 3) Минимизация тестовых векторов путем удаления повторяющихся входных последовательностей. Рис. 7 иллюстрирует таблицы процесса получения теста в соответствии с пунктами 2-3 алгоритма для функциональности $f(x) = \bar{x}_2\bar{x}_3 \vee x_1x_2x_3$, представленной схемной структурой:

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

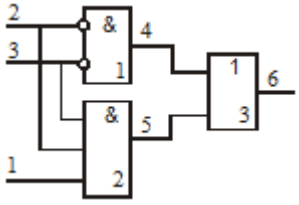

Рис. 7. Получение теста для схемной структуры булевой функции

Как альтернатива упомянутому выше, далее предлагается технологически 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-вектора 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.

Далее в таблице представлены результаты синтеза теста для функциональности, заданной уравнением: $f(x) = \bar{x}_2\bar{x}_3 \vee x_1x_2x_3$.

Данной логической функции от трех переменных соответствует 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
0	0	1	0	0	0	1	1
0	1	0	0	0	1	0	1
0	1	1	0	1	0	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	0
1	1	0	0	0	1	1	0
1	1	1	1	1	1	1	1

X_1	X_2	X_3	Y
0	1	1	0
1	1	1	1
0	1	0	0
0	0	0	1
0	0	1	0

С учетом выполнения пункта 3 алгоритм синтеза дает минимальный тест проверки входных переменных, который содержит 5 наборов, что представлено столбцом T, а также продублировано в явном виде правой таблицей.

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, не больше и не меньше.

4. Заключение

Ниже представлены формулировки научной новизны и практической значимости описанных исследований.

1) Впервые разработан метод и секвенсор безусловного синтеза тестов для функциональных логических компонентов, который характеризуется параллельным выполнением регистровых логических операций (shift, or, not, nxor) над кубитным вектором и его производными, что дает возможность существенно уменьшить время генерирования входных наборов и тестирования устройства в режиме embedded online.

2) Впервые разработан метод взятия производных для генерации тестов функциональных компонентов, который характеризуется параллельным выполнением регистровых логических операций (shift, or, not, nxor) над кубитным вектором, что дает возможность существенно уменьшить время генерирования входных наборов и тестирования устройства за счет аппаратной избыточности.

3) Практическая значимость исследований заключается в возможности облачной реализации быстродействующего метода синтеза тестов и моделирова-ния неисправностей для функциональных логических компонентов на основе параллельного выполнения регистровых логических операций (shift, or, not, nxor) над кубитным вектором и его производными, что дает возможность генерировать входные наборы и оценивать их качество в режиме online. Кроме того, облачный микросервис синтеза тестов и моделирования дефектов для функциональных логических компонентов может быть востребован для учебных и научных целей в процессах синтеза и анализа цифровых архитектур.

4) 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.

5) Дальнейшие исследования в данной области будут направлены на создние 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.

Литература: 1. *Рябцев В.Г., Муамар Д.Н.* Метод и средства визуализации алгоритмов тестов диагностирования запоминающих устройств // Электронное моделирование 2010. Т. 32, № 3. С. 43-52. **2.** *Zorian Y. , Shoukourian S.* Test solutions for nanoscale Systems-on-Chip: Algorithms, methods and test infrastructure // Ninth International Conference on Computer Science and Information Technologies Revised Selected Papers, Yerevan, 2013. P. 1-3. doi: 10.1109/CSITechnol.2013.6710371. **3.** *Tshagharyan G., Harutyunyan G., Shoukourian S. and Zorian Y.* Overview study on fault modeling and test methodology development for FinFET-based memories // 2015 IEEE East-West Design & Test Symposium (EWDTS), Batumi, 2015. P. 1-4. doi: 10.1109/EWDTS.2015.7493149. 4. *Проектирование* и тестирование цифровых систем на кристаллах / В. И. Хаханов, Е. И. Литвинова, И. В. Хаханова, О. А. Гузь. Харьков : ХНУРЭ, 2009. 484 с. **5.** *Abramovici M., Breuer M.A. and Friedman A.D.* Digital systems testing and testable design.- Computer Science Press. 1998. 652 p. **6.** *Кубитные* структуры данных вычислительных устройств / В. И. Хаханов, Ваджеб Гариби, Е. И. Литвинова, А. С. Шкиль // Электронное моделирование. 2015. Т. 37, № 1. С. 76-99. **7.** *Кубитные* технологии анализа и диагностирования цифровых устройств / В. И. Хаханов, Тамер Бани Амер, С. В. Чумаченко, Е. И. Литвинова // Электронное моделирование. 2015. Т. 37, № 3. С. 17-40. **8.** *Автоматизированное* проектирование цифровых устройств / С.С.Бадулин, Ю.М.Барнаулов и др./ Под ред. С.С. Бадулина. М.: Радио и связь. 1981. 240 с. **9.** *Michael A. Nielsen & Isaac L. Chuang.* Quantum Computation and Quantum Information. Cambridge University Press. 2010. 676 p. **10.** *Mikio Nfrfhara.* Quantum Computing. An Overview. Higashi-Osaka: Kinki University, 2010. 53p. **11.** *Курош А.Г.* Курс высшей алгебры. М.: Наука. 1968. 426с. **12.** *Бондаренко М.Ф., Хаханов В.И., Литвинова Е.И.* Структура логического ассоциативного мультипроцессора // Автоматика и телемеханика. 2012. № 10. С. 71-92.**13.** *Molnar L. and Gontean A.* Fault simulation methodes // 2016 12th IEEE International Symposium on Electronics and Telecommunications (ISETC), Timisoara, Romania, 2016. P. 194-197. **14.** *Hadjitheophanous S., Neophytou S.N. and Michael M.K.* Scalable parallel fault simulation for shared-memory multiprocessor systems // 2016 IEEE 34th VLSI Test Symposium (VTS). Las Vegas NV. 2016. P. 1-6.

15. *Pomeranz Irith*, Reddy Sudhakar M. Aliasing Computation Using Fault Simulation with Fault Dropping // IEEE Transactions on Computers. 1995. P. 139-144. **16.** *Ubar R., Kusaar J., Gorev M. and Devadze S.* "Combinational fault simulation in sequential circuits," 2015 IEEE International Symposium on Circuits and Systems (ISCAS), Lisbon.2015. P. 2876-2879. **17**. *Gorev M., Ubar R. and Devadze S.* "Fault simulation with parallel exact critical path tracing in multiple core environment," 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble. 2015.P. 1180-1185. **18.** *Pomeranz I.* "Fault simulation with test switching for static test compaction," 2014 IEEE 32nd VLSI Test Symposium (VTS), Napa, CA. 2014. P. 1-6. **19.** *Mirkhani S. and Abraham J. A.* "EAGLE: A regression model for fault coverage estimation using a simulation based metric". 2014 International Test Conference, Seattle, WA. 2014. P. 1-10.

Поступила в редколлегию 17.06.2016

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

Tamer Bani Amer, аспирант ХНУРЭ. Научные интересы: квантовые вычисления, тестирование и диагностика цифровых систем. Адрес: Украина, 61166, Харьков, пр. Науки, 14, тел. +3805770-21-326.

Емельянов Игорь Валерьевич, н.с. кафедры АПВТ ХНУРЭ. Научные интересы: квантовые вычисления, тестирование и диагностика цифровых систем. Адрес: Украина, 61166, Харьков, пр. Науки, 14, тел. + 3805770-21-326.

Любарский Михаил, соискатель кафедры АПВТ ХНУРЭ. Научные интересы: квантовые вычисления, тестирование и диагностика цифровых систем. Адрес: Украина, 61166, Харьков, пр. Науки, 14, тел. +3805770-21-326.

Хаханов Владимир Иванович, декан факультета КИУ ХНУРЭ, д-р техн. наук, проф. кафедры АПВТ ХНУРЭ,IEEESeniorMember, IEEE Computer Society Golden Core Member. Научные интересы: техническая диагностика цифровых систем, сетей и программных продуктов. Увлечения: баскетбол, футбол, горные лыжи. Адрес: Украина, 61166, Харьков, пр. Науки, 14, тел. +3805770-21-326. E-mail: hahanov@icloud.com.

УДК681.326:519.713

ИНФРАСТРУКТУРА ПРОЕКТИРОВАНИЯ SoC ДЛЯ МЕТОДА МУЛЬТИВЕРСНОГО СИНТЕЗА

ОБРИЗАН В.И.

Предлагается программно-аппаратная реализация моделей, методов и структур данных для проектирования цифровых систем на кристаллах, которая включает процедуры создания спецификации, синтеза, тестирования, моделирования и верификации на основе инфраструктуры, учитывающей промышленные средства компаний Aldec и Xilinx. Рассматриваются вопросы тестирования программных продуктов на реальных цифровых проектах создания IP-Core как примитивов для реализации цифровых систем на кристаллах.

Введение. Общая характеристика исследования

Цель — разработка и тестирование инфраструктуры проектирования цифровых систем на кристаллах, которая характеризуется параллельным выполнением мультиверсного синтеза функциональности, обеспечивающей существенное уменьшение времени создания проекта в условиях ограничения на аппаратные затраты.

Задачи:

1. Разработка метода мультиверсного синтеза управляющих и операционных автоматов в заданной инфраструктуре проектирования, ориентированных на архитектурные решения в метрике, минимизирующего время выполнения функциональности за счет распараллеливания операций при ограничении на аппаратные затраты.

2. Программная реализация моделей и методов мультиверсной разработки операционных устройств в рамках интегрированной системы проектирования функциональных и архитектурных решений SoC на основе использования продуктов верификации и синтеза компаний Aldec и Xilinx.

3. Тестирование и верификация программных модулей инфрастурктуры проектирования цифровых систем на кристаллах, а также определение эффективности предложенных моделей, методов и структур данных при создании реальных компонентов цифровых изделий.

1. Организация системы автоматизированного проектирования

По своей сути программа синтеза C++ в VHDL является машиной по трансформации исходного описания в результирующее. Таких трансформаций происходит несколько. Входная модель представлена на языке C++. Это алгоритмическое описание, которое решает поставленную перед инженером задачу.

Первый этап преобразования – *синтаксический анализ*. На этом этапе во входном описании из потока символов выделяются лексические элементы: ключевые слова, операторы и лексемы. В результате этого этапа получается синтаксическая модель исходного алгоритмического описания.

Второй этап преобразования – *трансформации на уровне синтаксической модели*. Они применяются для получения моделей с меньшим количеством состояний и занимающих меньше аппаратных ресурсов. К таким трансформациям относятся: вычисление констант, удаление недостижимого кода, встраивание функций, операции над циклами (развертки, свертки, распараллеливания).

Третий этап преобразования – *построение граф-схемы алгоритма*. В этой модели алгоритм представлен в виде отношений вершин двух типов: операций (арифметических, логических или ввода/вывода) и ветвлений.