УДК 681.5

С. Г. Мосин, **А. А. Кряжев**

Владимирский государственный университет
им. А. Г. и Н. Г. Столетовых
ул. Горького, 87, Владимир, 600000, Россия

E-mail: smosin@vlsu.ru

О ГЕНЕРАЦИИ ДИАГНОСТИЧЕСКИХ ТЕСТОВ НА ОСНОВЕ ТАБЛИЦ ТРАССИРОВОК

Предложен алгоритм генерации диагностических тестов, являющийся модификацией *D*-алгоритма. Исследованы существующие методы генерации диагностических тестов для цифровых электронных схем, выявлены недостатки. Предложен способ упрощения и распараллеливания алгоритма. Проведены экспериментальные исследования.

Ключевые слова: диагностика неисправностей, генерация тестов, *D*-алгоритм, таблицы трассировки, производительность алгоритма.

Введение

Развитие интегральных технологий, непрерывное увеличение размерности и сложности электронных схем, повышенные требования к качеству и надежности их функционирования определяют потребность в совершенствовании традиционных методов тестирования и диагностики изделий микроэлектроники. В эпоху интенсивного развития вычислительных технологий при решении задачи генерации контролирующих и диагностических тестов для электронных схем огромное значение имеют средства автоматизации процесса формирования входных тестовых последовательностей.

На практике наибольшее распространение получили методы диагностирования на основе процедуры активизации пути [1], разновидностью которых является *D*-алгоритм [2; 3]. Однако *D*-алгоритм имеет ряд недостатков: затратные символьные вычисления и анализ значений; необходимость хранения состояний внутренних линий в виде символов; *D*-кубы, используемые в алгоритме, содержат избыточную информацию.

В работе предложена модификация *D*-алгоритма, в которой устранены выявленные недостатки классического алгоритма, адаптированная для использования на современных вычислительных системах.

Модифицированный алгоритм

В отличие от *D*-алгоритма, где используются *D*-кубы, в модифицированном алгоритме предложено использовать таблицы трассировки, являющиеся, по сути, *D*-кубами максимальной кратности. Это означает, что алгоритм работает как с одиночными, так и кратными путями. Сравнение этих понятий представлено в табл. 1.

Модифицированный алгоритм включает 3 базовые процедуры.

1. Установка значения неисправной линии. Значение на линии устанавливается равным инверсии неисправного значения (противоположным значению константной неисправности).

2. Операция установки. В ходе операции устанавливаются значения линий в направлении от выходов выбранного элемента к первичным входам (входам схемы). Выбранным считает-

Таблица 1

Сравнение понятий *D*-куба и таблиц трассировки

	***D*-кубы**	**Таблицы трассировки**
Множество элементов	$\{0, 1, x, d, \overline{d}\}$	$\{0, 1, t, \overline{t}\}$
Исходные данные для построения	Вырожденное покрытие	Таблица истинности
Процедура построения	Попарное пересечение строк	
Правила построения	$0 \& 0 = 0 \& x = x \& 0 = 0$ $1 \& 1 = 1 \& x = x \& 1 = 1$ $x \& x = x$ $1 \& 0 = d$ $0 \& 1 = \overline{d}$	$0 \& 0 = 0$ $1 \& 1 = 1$ $1 \& 0 = t$ $0 \& 1 = \overline{t}$
Пример для элемента 2-*ИЛИ*	Кратность 1: $d\,0\,d$ / $0\,d\,d$ Кратность 2: $d\,0\,d$ / $0\,d\,d$ / $d\,d\,d$	$t\,0\,t$ / $0\,t\,t$ / $t\,t\,t$ На практике используется сокращенная таблица для конкретной трассируемой линии. Если это первая линия, то таблица принимает вид выделенных ячеек.
Использование	Операция *D*-пересечения; Операция *D*-установки (используются *D*-кубы неисправности)	Операция трассировки

ся элемент, у которого рассматриваемая линия является входом. Установка происходит по следующим правилам:

2.1. Выбрать запись из таблицы истинности, для которой значения для входов элемента равны уже установленным значениям линий.

2.2. Установить значения остальных входов:

а) применить операцию установки на элементы, выходы которых являются вновь установленными входами, ранжируя по критерию близости к первичным входам;

б) если операция достигла первичных входов, то выполнение считается успешным;

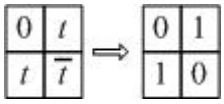
в) если значения всех записей вырожденного покрытия противоречат уже установленным линиям, то происходит возврат на предыдущий элемент и выполняются действия, начиная с п. 2.1.

В *D*-алгоритме этой операции соответствуют операции импликации и доопределения.

3. Операция трассировки. В ходе операции происходит перенос значений от входа выбранного элемента к выходам схемы.

3.1. Выбрать запись из таблицы трассировки, для которой значения для входов и выходов элемента равны уже установленным значениям линий.

3.2. Установить значения остальных входов и выходов по правилу: если для устанавливаемой линии значение равно t, то ей присваивается значение трассируемой линии, если же значение равно $\overline{t}$, то ей присваивается значение, противоположное значению трассируемой линии. Например, если значение трассируемой линии t равно 1, то таблица трассировки будет преобразована следующим образом:



Необходимо:

а) применить операцию установки на текущий элемент для доустановки всех входов;

б) применить операцию трассировки на элементы, входы которых являются вновь установленными выходами, ранжируя по критерию близости к выходам схемы;

в) если в процессе операции были достигнуты выходы схемы, то выполнение считается успешным;

г) если значения всех записей таблицы трассировки противоречат уже установленным линиям, то происходит возврат на предыдущий элемент и выполняются действия, начиная с п. 3.1.

В *D*-алгоритме этой операции соответствует операция *D*-прохода.

Таким образом, линии схемы принимают значения 0 или 1 в отличие от *D*-алгоритма, где хранятся значения 0, 1, *d*.

Результатом *D*-алгоритма являются значения входов схемы, необходимые для выявления неисправности, а также значения на выходах *d* или $\overline{d}$. При этом символ *d* означает, что элемент или схема в исправном состоянии на выходе должен иметь 1, в неисправном – 0. Для $\overline{d}$ соответственно наоборот. В модифицированном алгоритме значение на выходе является значением для исправной линии схемы, для неисправной линии значение инвертируется.

В модифицированном алгоритме по сравнению с *D*-алгоритмом была упрощена операция переноса значения через элемент от входов к выходам (операция *D*-пересечения), где пересекаются 2 множества: уже определенных значений линий и *D*-куб элемента, а затем проводится анализ значений и их возможная корректировка в зависимости от полученных значений λ и μ. В предложенном алгоритме участвуют только подмножество выводов элемента и его сокращенная таблица трассировки. Анализ сводится к сравнению значений нулей и единиц.

Особенности реализации модифицированного алгоритма

Реализация предложенного алгоритма выполнена на языке Python 3.2 с использованием возможности генераторов. Генератор – функция, которая может многократно вызываться, сохраняя контекст выполнения. Генераторы использовались для продолжения работы после неудачной операции установки или трассировки при возврате на предыдущий элемент.

Хранение массива целых чисел требует меньше памяти, чем при хранении символьных структур данных.

Операция трассировки быстрее операции *D*-прохода, поскольку анализ сводится к сравнению целочисленных числовых значений, представленных в виде нулей и единиц.

Применительно к архитектурам современных ВС в программной системе была реализована поддержка многопроцессорности / многопоточности благодаря использованию futures-примитивов.

Последовательный код:

```
for r in map(trace_fault, faults_data): #log
```

Параллельный код:

```
with ProcessPoolExecutor(parallel_num) as executor:
    for r in executor.map(trace_fault, faults_data): #log
```

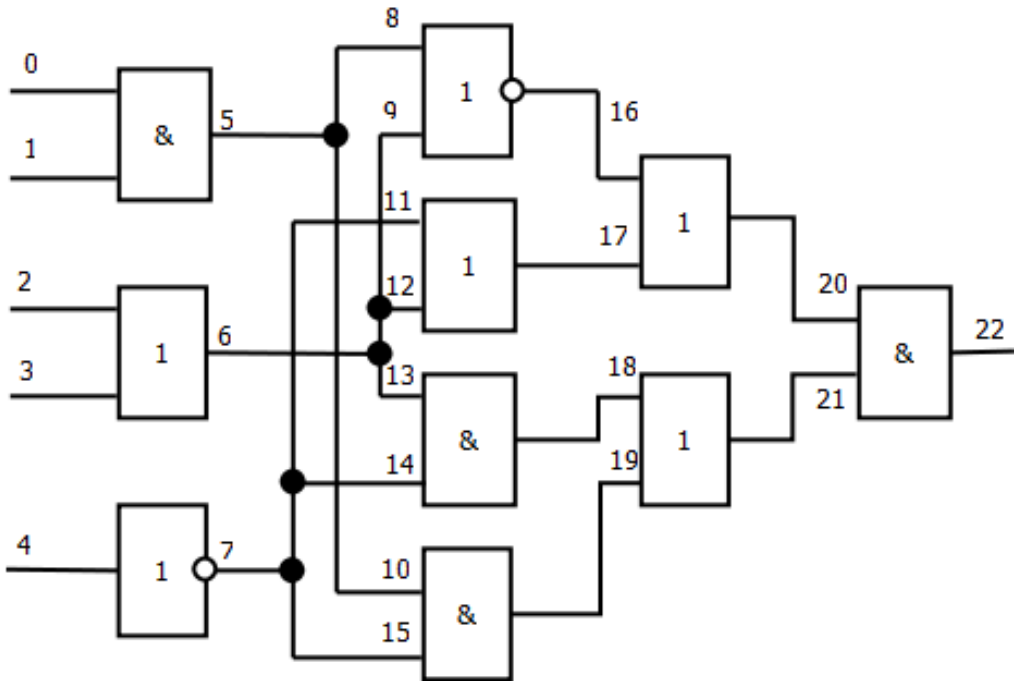
Результатом выполнения модифицированного алгоритма является кортеж

[(*f*, *v*) ins: {*in*0: *i*1, *in*1: *i*1, …, *in*i: *i*i.}, outs: {*out*0: *o*0, …, *out*j: *o*j}],

где f – номер внутренней линии, v – значение константной неисправности, {*in*i, *i*i} – номер i-го первичного входа схемы и соответствующее ему значение входного тестового сигнала, {*out*j: *o*j} – номер j-го первичного выхода и соответствующее значение тестового отклика схемы.

Результаты экспериментальных исследований

Сравнение двух алгоритмов выполнено на комбинационной схеме для константных неисправностей на внутренних линиях:



Допустим, что в схеме присутствует константная неисправность: обрыв линии 18 (18, 0).

Ход выполнения алгоритма для данного примера.

1. Установить значение линии 18 равным единице.

2. Применить операцию установки, в ходе которой задать значения вначале для линий 13, 14, 9, 12, 11, 15 равными единицам, затем установить значения второй линии равным нулю, третьей линии – равным единице и значение четвертой линии – равным нулю. На этом операция установки завершается, поскольку достигнуты первичные входы.

3. Применить операцию трассировки на выбранный элемент, вход которого линия 18. Таким образом, установлено значение линии 19, равное нулю, и значение линии 21, равное единице.

4. Применить операцию установки на текущий элемент, в результате которой установить значения линий 10, 15 и далее в направлении первичных входов.

5. Применить операцию трассировки на элемент, входом которого является линия 21. И так далее.

Результат выполнения алгоритма для данной неисправности – следующая запись:

(18, 0) ins: {0: 0, 1: 0, 2: 0, 3: 1, 4: 0}, outs: {22: 1}

Это означает, что для выявления данной неисправности необходимо на первичные входы 0 – 4 подать тестовый набор:

{0: 0, 1: 0, 2: 0, 3: 1, 4: 0}

Таблица 2

Результаты тестирования производительности алгоритма

Количество схем	Время выполнения, мс		Коэффициент прироста производительности
	D-алгоритм	Предложенный алгоритм	
1	Последоват.: 0.142	Последоват.: 0.0434 4 потока: 0.0530 4 процесса: 0.1928	3.27 2.68 0.74
100	Последоват.: 11.97	Последоват.: 4.35 4 потока: 4.69 4 процесса: 1.40	2.75 2.55 8.55
500	Последоват.: 59.68	Последоват.: 21.70 4 потока: 23.48 4 процесса: 6.59	2.76 2.54 9.06

На выходе схемы значение, равное 1, означает, что при данном наборе корректное значение выхода схемы равно 1, в неисправном состоянии – 0.

Для наибольшей достоверности результата тесты проводились на различном количестве схем, моделируемых последовательно. Данные эксперимента приведены в табл. 2.

Анализ работоспособности алгоритма проводился на стандартном наборе тестовых схем ISCAS'85 [4]. Результаты экспериментов для схемы C432:

- для *D*-алгоритма время выполнения составило 11460,0276 с, среднее время обработки неисправности – 9,1242 с;
- время выполнения последовательного модифицированного *D*-алгоритма – 3058,859 с, а среднее время обработки неисправности – 2,435 с, т. е. в 3,75 раза меньше, чем при использовании обычного *D*-алгоритма.

Заключение

Модифицированный *D*-алгоритм позволяет сократить временные затраты на процесс формирования входных тестовых наборов для электронных цифровых схем. По сравнению с последовательным *D*-алгоритмом использование предложенного алгоритма дало прирост производительности почти в 3 раза при последовательном выполнении программы и в 9 раз при параллельном выполнении программы. Данный эффект обеспечивает возможность формирования диагностических тестовых наборов для большего числа возможных неисправностей за фиксированное время или снижение временных и стоимостных затрат при работе с фиксированным множеством возможных неисправностей.

Список литературы

1. *Ланцов В. Н., Мосин С. Г.* Современные подходы к проектированию и тестированию интегральных микросхем: Моногр. Владимир, 2010. 285 с.

2. *Chang H. Y., Manning E., Metze G.* Fault Diagnosis of Digital Systems. Wiley-Interscience, 1970. 232 p.

3. *Roth J. P.* Diagnosis of Automata Failures: A Calculus and a Method // IBM Journal of Research and Development. 1966. No. 10. P. 278–291.

4. *Hansen M. C., Yalcin H., Hayes J. P.* Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering // IEEE Design and Test. 1999. No. 16. P. 72–80.

Материал поступил в редколлегию 15.05.2012

S. G. Mosin, A. A. Kryazhev

A GENERATION OF DIAGNOSTICS TESTS BASED ON THE TRACING TABLES

The algorithm of generating diagnostics tests as modification of D-algorithm has been proposed. The existent methods of diagnostics tests generation for digital electronic circuits have been investigated and disadvantages identified. The way for algorithm simplification and parallelization has been proposed. The experimental study has been done.

Keywords: fault diagnosis, test generation, *D*-algorithm, tracing tables, algorithm performance.