УДК 681.518.54

А.В. Борисевич, студент,
Р.О. Берзин, студент
Севастопольский национальный технический университет
Студгородок, г. Севастополь, Украина, 99053
E-mail: xmastal@mail.ru

ГЕНЕТИЧЕСКИЙ АЛГОРИТМ ДЛЯ ПОСТРОЕНИЯ ДИАГНОСТИЧЕСКОГО ТЕСТА СЛОЖНЫХ ЦИФРОВЫХ СХЕМ

В работе рассматривается решение задачи синтеза диагностического теста для синхронных цифровых схем, основанное на применении генетического алгоритма. Определяются основные операции генетического алгоритма над тестовыми последовательностями. Формулируются требования к тестируемой схеме.

Синтез диагностических тестов для цифровых устройств является важнейшей задачей при обслуживании и ремонте схем, а также для наладки на этапе изготовления. Для простоты ограничимся моделью однократных константных неисправностей на полюсах схемы.

Будем считать, что тестовая последовательность T диагностирует ошибки E_i и E_j в случае, если после применения хотя бы одного вектора из T получены различные значения выходных сигналов устройства. Если такая последовательность T существует, то неисправности E_i и E_j различимы (диагностируемы). В противном случае неисправности E_i и E_j неразличимы (недиагностируемы). Множество ошибок $\{E_k\}$ называется множеством неразличимых ошибок, если любая пара $E_i, E_j \in \{E_k\}$ неразличима.

Общепринятым методом диагностики является двухуровневая (или в общем случае многоуровневая диагностика). Суть метода состоит в том, что применение некоторого теста T к устройству разбивает $\{E_k\}$ на классы в зависимости от полученной реакции устройства с ошибкой. Первоначально множество неразличимых ошибок содержит все ошибки устройства. Пусть некоторая последовательность T_0 разбивает это множество на классы. Далее к каждому классу также применяются некоторые последовательности T_j, которые продолжают разбиение. Процесс разбиения продолжается, пока в каждом *k*-ом еще неразбитом классе не останется по одной ошибке, что соответствует однозначному диагнозу: $|\{E_k\}| = 1$. В действительности добиться выполнения этого требования очень трудно, если речь идет о схемах большой сложности. Далее предлагается оценка качества теста следующего вида:

$$\mu = \frac{N}{\sum_{k=1}^{N} |\{E_k\}|}. \qquad (1)$$

В (1) суммирование производится по всем полученным N неразбитым классам неразличимых ошибок.

Построение диагностических тестов представляет собой сложную и трудоемкую в вычислительном смысле проблему. В качестве примера детерминированного алгоритма построения диагностических тестов можно привести работу [1]. Однако на практике предложенные решения редко применяются из-за следующих двух основных причин: многие схемы не могут быть адекватно описаны моделью классического конечного автомата (например, схема, в которой действуют несколько тактовых сигналов и некоторые из них формируются внутри нее), а также вследствие большой вычислительной сложности алгоритмов.

Предложенный нами алгоритм и его реализация основан на работах [2,3], в которых для поиска проверяющего теста используется генетический алгоритм. Модификации этих методов позволяет использовать генетический алгоритм для более сложной задачи – построения диагностического теста.

Генетические алгоритмы [4] представляют собой широко применяемый класс алгоритмов случайного поиска экстремума функций, дающие значительное сокращение перебора и повышение быстродействия по сравнению с другими методами оптимизации. Далее при рассмотрении метода мы будем пользоваться терминологией, приведенной в [4].

Пусть схема представлена в виде совокупности N_e логических элементов фиксированного типа (НЕ, И, И-НЕ и т.д.) и N_d триггеров также определенного типа (D, JK и т.д). В случае, если используются более сложные компоненты, они должны быть декомпозированы в схему из логических вентилей и триггеров.

Пусть задано множество неисправностей схемы $\aleph = \{E_k\}$. Его можно рассматривать как начальный класс неразличимых ошибок. Алгоритм построения теста состоит из двух этапов:

1) построение начальной популяции $T^{нач} = \{T_1, \ldots T_q\}$ тестовых последовательностей и выбор класса $\aleph_j \subseteq \aleph$ неразличимых ошибок для разбиения;

2) построение тестовой последовательности T^*, которая разбивает $\aleph_j$ на два или более класса.

Фактически генетический алгоритм применяется только на втором этапе. Первый этап реализует случайную генерацию $T^{нач}$ и разбиение классов с помощью полученных последовательностей (выявление «про-

стых» ошибок). Как только обнаружен класс, который не разбивается ни одной последовательностью из $T^{нач}$, происходит переход ко второму этапу, в котором с помощью генетического алгоритма находится последовательность T^* или определяется ее отсутствие (если после заданного числа поколений класс $\aleph_j$ так и не будет разбит). После возврата на первый этап алгоритма, популяция $T^{нач}$ заново не генерируется, а является последним поколением последовательностей, полученных в результате работы генетического алгоритма. Процесс завершается, если оценка теста по (1) удовлетворяет $\mu \le \mu_0$ или произведено заранее заданное число итераций.

Применение генетического алгоритма для решения задачи поиска требует определения целевой функции $F(T)$. Целевая функция $F(T)$ должна в числовом виде выражать приближение полученной последовательности T к решению.

Очевидно, что функция $F(T)$ должна выражать «активность» схемы – различие в сигналах между схемой с ошибкой $E_i \in \aleph_k$ и схемой с ошибкой $E_j \in \aleph_k$. Очевидно, что чем больше «активность», тем вероятнее тот факт, что применение теста T приведет к различным реакциям на выходах схем с разными ошибками. В общем случае можно записать:

$$F(T,\aleph) = \max_{\substack{E_i,E_j \in \aleph \\ i>j}} \left\{ \max_{v_k \in T} \left\{ A(E_i,E_j,T,v_k) \right\} \right\}. \qquad (2)$$

Здесь $A(E_i,E_j,T,v_k)$ – функция, выражающая разность внутренних сигналов схемы после применения векторов от v_0 до v_k из последовательности T. В выражении (2) оценка берется по максимуму $A(E_i,E_j,T,v_k)$, поскольку максимальная разность между двумя состояниями схем с ошибками E_i и E_j обеспечивает наиболее вероятное разбиение $\aleph$ на два класса, в одном из которых будет содержаться E_i, а в другом – E_j.

Разность состояний сигналов схемы можно оценить обычным образом с помощью взвешенной суммы

$$A(E_i,E_j,T,v_k) = C_d \sum_{q=1}^{N_d} w^d{}_q d^d{}_q(E_i,E_j) + C_e \sum_{q=1}^{N_e} w^e{}_q d^e{}_q(E_i,E_j).$$

Здесь C_d и C_e – весовые коэффициенты, выражающие важность состояния триггера или логического элемента соответственно;

$w^d{}_q$ и $w^e{}_q$ – веса q-го триггера и логического элемента соответственно;

$d^d{}_q(E_i,E_j)$ – функция определенная следующим образом:

$d^d{}_q(E_i,E_j) = \begin{cases} 0, \text{ триггер } q \text{ имеет одинаковое состояние в схемах с ошибкой } E_i \text{ и } E_j \\ 1, \text{ триггер } q \text{ имеет разное состояние в схемах с ошибкой } E_i \text{ и } E_j \end{cases}$;

$d^e{}_q(E_i,E_j)$ – функция определенная аналогично $d^d{}_q(E_i,E_j)$ для логических элементов.

Весовой коэффициент $w^d{}_q$ вычисляется следующим образом:

$w^d{}_q = \max\limits_{j\in 1,N_d}\left\{\left|P_{d-out_j}\right|\right\} - \left|P_{d-out_q}\right|$, где $\left|P_{d-out_j}\right|$ – длина пути от выхода триггера до выхода схемы или входа триггера (в элементах).

Аналогично для логического элемента $w^e{}_q = \max\limits_{j\in 1,N_e}\left\{\left|P_{e-out_j}\right|\right\} - \left|P_{e-out_q}\right|$.

Приведенная выше функция (2) позволяет оценить качество тестовой последовательности для расщепления некоторого класса $\aleph$ на подклассы. Для того, чтобы минимизировать длину тестовой последовательности можно предложить следующую функцию

$$F(T,\aleph) = \max_{\substack{E_i,E_j\in\aleph \\ i>j}}\left\{\max_{v_k\in T}\left\{\lambda^k \cdot A(E_i,E_j,T,v_k)\right\}\right\},$$

где λ – эмпирический параметр ($0 < \lambda < 1$).

В [2,3] было рассмотрено решение задачи генерации проверяющего теста для схем с одним тактовым входом. Однако многие схемы содержат несколько тактовых входов. В этом случае, простое применение генетического алгоритма для генерации значений для входов тактового сигнала крайне неэффективно. Для решение этой задачи разделим множества входов I следующим образом $I = I_{inf} \cup I_{clk} \cup I_{rst}$, где I_{inf} – информационные входы (входы со статическим управлением), I_{clk} – входы синхронизации (входы с динамическим управлением) и I_{rst} – входы сброса (входы асинхронной установки триггеров). Состояния информационных входов может принимать значения $I_{inf\,j} \in \{0,1\}$, в то время как входы синхронизации должны управляться событиями. Выделим два типа событий: импульс положительной полярности и логический 0, обозначив соответственно $I_{clk\,j} \in \{P,0\}$. Поскольку имеется всего два типа событий, то рабочими данными для генетического алгоритма по-прежнему являются строки из двоичных векторов. Значение 0 в векторе имеет прежний смысл, а значению 1 сопоставлен импульс на соответствующем тактовом входе. Если при моделировании тестового вектора на некотором тактовом входе появляется значение 1, то система моделирования заменяет данный вектор на три вектора с аналогичным содержимым, но с последовательностью значений тактового входа $(0,1,0)$.

Требованием к схеме является $I_{rst} \neq \emptyset$. Под входом сброса понимается такой вход схемы, значение которого управляет входами асинхронной установки триггеров схемы через тривиальную цепь (не содержащую логических элементов, или через цепь, которая может быть априорно протестирована). Тестируемой схеме сопоставлен вектор сброса v_{rst}, применение которого к входам I_{rst} переводит все триггеры устройства в состояния, не зависящие от их предыдущего значения.

Опишем преобразования, которые выполняются генетическим алгоритмом над тестовыми последовательностями.

Реализация кроссинговера $Cr(T_i, T_j)$ может быть вертикальной или горизонтальной. При модификации теста выбирается один из видов кроссинговера с равной вероятностью. Суть горизонтального кроссинговера состоит в том, что тесты T_i и T_j разбиваются на две части следующим образом: $T_i = \{v_0, \ldots v_k\} \cup \{v_{k+1}, \ldots v_{|T_i|-1}\}$, $T_j = \{v_0, \ldots v_q\} \cup \{v_{q+1}, \ldots v_{|T_j|-1}\}$, где индексы $k \in [0, |T_i| - 1]$ и $q \in [0, |T_j| - 1]$ – выбираются случайно. Результатом кроссинговера является последовательность $T = \{v_0, \ldots v_k\} \cup \{v_{q+1}, \ldots v_{|T_j|-1}\}$. Вертикальный кроссинговер выполняется следующим образом. Пусть даны две последовательности T_i и T_j . Выберем $L = \max\{|T_i|, |T_j|\}$. Далее осуществляется заполнение векторов результата скрещивания: $T = \{v_0, \ldots v_{L-1}\}$. Каждый компонент j вектора v_i равновероятно выбирается либо из соответствующего вектора T_i , либо T_j : $v_{i,j} \in \{v^{T_i}{}_{i,j}, v^{T_j}{}_{i,j}\}$. Если $i \geq \min\{|T_i|, |T_j|\}$, то значение отсутствующего в T_i или в T_j бита генерируется случайно.

Мутация $M(T_i)$ выполняется следующим образом: случайно выбирается n различных векторов из T_i (удобно положить $n = \gamma \cdot |T_i|$). Далее в данных векторах инвертируется m (аналогично, $m \sim |v_k|$) случайно выбранных битов.

Предложенный алгоритм прошел проверку на библиотеке схем ISCAS-89. Для всех схем теста получена оценка качества теста $0{,}75 \leq \mu \leq 0{,}95$, что говорит о высокой эффективности построения диагностических тестов. Несомненным достоинством алгоритма является применимость к широкому классу схем: комбинационным, синхронных с одним и несколькими тактовыми входами, а также к асинхронным. С другой стороны, этот метод прост как для понимания, так и для реализации, и его быстродействие полностью определяется быстродействием системы моделирования.

Дальнейшая работа будет направлена на исследование применения генетических алгоритмов совместно с традиционными детерминированными методами, изложенными в [1]. Также планируется обобщение изложенного метода на смешанные аналого-цифровые схемы.

Библиографический список

1. Волынский М.Б. Обнаружение и поиск неисправностей программируемых логических матриц / М.Б. Волынский, В.Г. Новоселов // Микроэлектроника. — 1983. — Т.12 — №4. — С. 306 – 312.

2. Rudnick E.M., Holm J.G., Saab D.G., Patel J.H., Application of Simple Genetic Algorithm to Sequential Circuit Test Generation // Proc. European Design & Test Conf.— 1994. — P.40 – 45.

3. P. Prinetto, M. Rebaudengo, M. Sonza Reorda, An Automatic Test Pattern Generator for Large Sequential Circuits based on Genetic Algorithms // Proc. Int. Test Conf. — 1994 — P.178 – 186.

4. Вороновский Г.К. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности / Г.К. Вороновский, К.В. Махотило, С. Н. Петрашев — Харьков: ОСНОВА, 1997. — 112 с.

Поступила в редакцию 27.09.2005 г.