УДК 681.32

Ю.А. СКОБЦОВ[1], В.Ю. СКОБЦОВ[2], Ш.Н. ХИНДИ[1]

Донецкий национальный технический университет, Украина[1]
Институт прикладной математики и механики НАН Украины, Украина[2]

ИЕРАРХИЧЕСКИЕ ЭВОЛЮЦИОННЫЕ АЛГОРИТМЫ ПОСТРОЕНИЯ ПРОВЕРЯЮЩИХ ТЕСТОВ ЦИФРОВЫХ ПОСЛЕДОВАТЕЛЬНОСТНЫХ СХЕМ

Представлен иерархический генетический алгоритм генерации тестов, где на нижнем уровне эволюционными методами сначала генерируются некоторые входные характеристические последовательности, которые позволяют установить некоторые элементы (прежде всего памяти) в определенные состояния и тем самым упростить генерацию тестов. Генетический алгоритм второго верхнего уровня при генерации тестов в качестве «строительного материала» использует произвольные входные последовательностями и характеристические последовательности, построенные на нижнем уровне ГА, что делает эволюционный поиск более направленным и повышает его эффективность. При оценке полноты тестов используется кратная стратегия наблюдения сигналов.

последовательностные схемы, генетические алгоритмы, генерация тестов, кратная стратегия наблюдения

Введение

Эффективность систем генерации проверяющих тестов для цифровых систем в значительной степени зависит от применяемых методов и алгоритмов построения проверяющих тестов. В настоящее время при генерации тестов широко применяются последовательные и параллельные генетические алгоритмы (ГА), которые позволили существенно повысить их качество и быстродействие [1,2].

Целью данной работы является дальнейшее развитие эволюционных алгоритмов построения проверяющих тестов для схем с памятью за счет использования иерархических генетических алгоритмов.

Кратная стратегия наблюдения сигналов

Постановка задачи построения проверяющего теста для заданной неисправности в последовательностной схеме существенно зависит от используемой стратегии наблюдения неисправностей [1]. Пусть исправная последовательностная схема реализует конечный автомат $A = (Y, X, Z, \delta, \lambda)$, где Y, X, Z - конечные множества состояний, входных и выходных сигналов соответственно; $\delta : Y \times X \to Y$ - функция переходов, определяющая следующее состояние автомата; $\lambda : Y \times X \to Z$ - функция выхода, определяющая выходной сигнал. Функции δ и λ реализуются комбинационными схемами, где :

$$Y = (y_1, ..., y_k),\ \text{ãääå}\ y_i = (0,1)\ \text{äëÿ}\ i = \overline{1,k}\ ; \qquad (1)$$

$$X = (x_1, ..., x_n),\ \text{ãääå}\ x_j = (0,1)\ \text{äëÿ}\ j = \overline{1,n}\ ; \qquad (2)$$

$$Z = (z_1, ..., z_m),\ \text{ãääå}\ z_j = (0,1)\ \text{äëÿ}\ j = \overline{1,m}\ . \qquad (3)$$

Пусть $X(1)$, $X(2)$, …, X(p) – входная последовательность длины p. Тогда $Y(Y_0,0)$, $Y(Y_0,1)$,…, $Y(Y_0,p)$ – последовательность состояний автомата, которую он проходит из начального состояния $y_0 \in Y$ под воздействием входной последовательности $X(1)$, $X(2)$,…, $X(p)$. Пусть $Z(Y_0,0)$, $Z(Y_0,1)$,…, $Z(Y_0,p)$ – обозначает выходную последовательность, производимую автоматом из начального состояния Y_0 при подаче входной последовательности $X(1)$, $X(2)$,…, $X(p)$. Обозначим через $z_j(y_0,k)$ для $j = \overline{1,m}$ значение j-го выхода на k-м шаге моделирования. Используя эти обозначения, следующее состояние определяется следующим образом:

$$Y(y_0, t) = \begin{cases} Y_0\ \text{äëÿ}\ t = 0 \\ \delta(X(t), Y(Y_0, t-1))\ \text{äëÿ}\ t \neq 0 \end{cases}. \qquad (4)$$

Аналогично выход $Z(y_0,k)$ определяется функцией λ.

Неисправность f преобразует автомат A в $A=(Y,X,Z,\delta^f,\lambda^f)$, где функции δ^f,λ^f состояния Y^f и Z^f определяются таким же образом. Далее расмотрим различные стратегии обнаружения неисправностей в последовательностных схемах.

Определение 1. Одиночная константная неисправность f называется обнаружимой в последовательностной схеме входной последовательностью $X(1),X(2),\ldots,X(p)$ относительно стратегии одиночного наблюдения выходов, если

$$\exists b\in\{0,1\},\exists t\le p,\exists i\le k,\forall(r,q):\\ ((z_i(r,t)=b)\wedge(z_i^f(q,t)=\bar{b})), \qquad (5)$$

где r – начальное состояние исправной схемы и q – начальное состояние неисправной схемы.

Это определение говорит, что при данной стратегии неисправность считается обнаружимой, если найдется (по крайней мере один) момент времени (такт) t такой, что для любой пары состояний (r,q) исправной и неисправной схем некоторый i-й выход z_i имеет различные значения в исправной и неисправной схеме. Ключевым моментом является то, что любая пара состояний исправной и неисправной схемы должна выдать различные выходные реакции в один и тот же такт времени. Для последовательностных схем иногда используется другая стратегия кратного наблюдения, при которой различные пары состояний исправной и неисправной схем могут различаться в различные такты (моменты) времени.

Определение 2. Одиночная константная неисправность f называется обнаружимой в последовательностной схеме входной последовательностью $X(1), X(2),\ldots, X(p)$ относительно стратегии кратного наблюдения выходов, если:

$$\forall(r,q)\exists t\le p,\exists i\le k,\exists b\in\{0,1\}:\\ ((z_i(r,t)=b)\wedge(z_i^f(q,t)=\bar{b})). \qquad (6)$$

Отметим, что принципиальное отличие между этими стратегиями состоит в следующем. Согласно первой стратегии неисправность обнаружима и, следовательно, для нее можно построить тестовую последовательность) если найдется один момент времени t, такой что независимо от начального состояния исправной и неисправной схем значения выходных сигналов различны для исправной и неисправной схем. То есть все пары состояний исправной и неисправной схемы выдают различные выходные сигналы в один и тот же момент времени. Согласно второй стратегии для каждой пары состояний исправной и неисправной схемы существует свой момент времени, в котором они дают различные выходные сигналы.

Таким образом, при построении теста для схем с памятью необходимо найти входную последовательность $X(1)$, $X(2),\ldots$, $X(p)$, для которой выполняется (5) или (6) в зависимости от применяемой стратегии наблюдения выходов. Естественно вторая стратегия требует больших вычислительных ресурсов, но позволяет повысить полноту тестов.

Как правило, при построении тестов для последовательностных схем применяется стратегия одиночного наблюдения выходных сигналов, где любая пара состояний исправной и неисправной схемы должна выдать различные выходные реакции в один и тот же такт времени (обычно последний). В отличие от разработанных ранее методов в данной работе используется кратная стратегия наблюдения выходных сигналов, при которой различные пары состояний исправной и неисправной схем могут различаться в различные такты (моменты) времени, что позволяет повысить полноту тестов, но требует больших вычислительных ресурсов.

Двухуровневый ГА

При построении тестов для последовательностных схем целесообразно использовать иерархический подход, который может быть реализован на структурном уровне следующим образом. Построе-

ние тестовой последовательности для такой неисправности выполняется в два этапа: 1) активизация неисправности, 2) распространение влияния неисправности, как это показано на рис. 1. При этом используется структурная модель в виде итеративной комбинационной схемы [1].

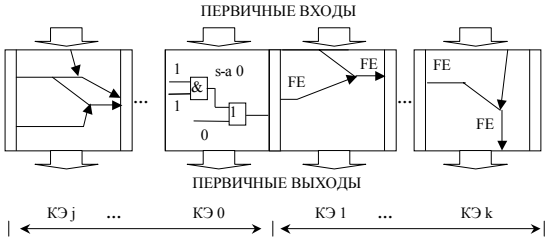

Рис. 1. Два этапа генерации тестовой последовательности

В этом случае на нижнем уровне эволюционными методами сначала генерируются некоторые входные характеристические подпоследовательности, которые позволяют установить значения сигналов в определенные значения для некоторых (прежде всего) элементов и тем самым упростить генерацию тестов. При этом используются генетические алгоритмы нижнего уровня, где в качестве особей выступают входные подпоследовательности, задаваемые двоичными таблицами [1]. Применение многозначных алфавитов моделирования в процессе их генерации позволяет повысить эффективность.

Генетический алгоритм второго верхнего уровня при генерации тестов как структурные блоки использует входные последовательностями и характеристические последовательности, построенные на нижнем уровне ГА, что делает эволюционный поиск более "направленным" и повышает его эффективность.

Характеристические последовательности

На нижнем уровне с помощью ГА генерируются характеристические подпоследоватедьности, которые устанавливают значения сигналов для некоторых элементов в определенные значения. Для построения тестов (на верхнем уровне) полезными являются следующие входные характеристические последовательности.

1. *S-последовательности*. Назовем последовательность S_i , устанавливающей *i*-й триггер, входную последовательность, которая устанавливает *i*-й триггер в единичное состояние.

2. *R-последовательности*. Аналогично назовем последовательность R_i , сбрасывающей (очищающей) *i*-й триггер, входную последовательность, которая сбрасывает *i*-й триггер в нулевое состояние.

Будем называть последовательность S_i (R_i) типа *A*, если она позволяет установить (сбросить) *i*-й триггер из полностью неопределенного состояния, когда все триггера имеют неопределенные значения. Обычно длина этих входных последовательностей ограничивается 4*D*, где *D* - структурная последовательностная глубина [1]. Эти же последовательности будем называть типа *B*, если они позволяют установить триггера в соответствующее значение при условии, что в начальном состоянии некоторые триггера имеют некоторые определенные (0,1) значения. Последовательности типа B генерируются при необходимости в процессе построения тестовой последовательности. Под псевдорегистром в дальнейшем мы будем понимать группу триггеров. Кроме описанных выше последовательностей, часто также используются последовательности, которые позволяют установить псевдорегистр в некоторое определенное состояние (т.е. одновременно установить несколько триггеров в заданные состояния).

3. *Различающие последовательности*. Различающая последовательность типа A для i-го триггера определяется как входная последовательность, которая производит различные выходные реакции исправной схемы для двух различных состояний, отличающихся значением сигнала на i-ом триггере при неопределенных значениях остальных триггеров, как это показано на рис. 2.

Триггера		Триггера		Триггера	
Исправная схема	uuuu1uuu	Исправная схема	uuuu1uuu	Исправная схема	uuuu1Suu
Исправная схема	uuuu0uuu	Неисправная схема	uuuu0uuu	Неисправная схема	uuuu0Suu
а) тип А		б) тип Б		в) тип В	

Рис. 2. Типы различающих последовательностей

Различающей последовательностью типа *B* для *i*-го триггера называется входная последовательность, производящая различные выходные реакции исправной и неисправной схем для состояний, отличающихся значением сигнала на i-ом триггере при неопределенных значениях остальных триггеров, как это показано на рис.2. Различающая последовательность типа *C*, показанная на рис. 2, аналогична последовательности типа *B* за исключением того, что некоторое подмножество триггеров имеет определенные значения (0,1 а не u), обозначенные *S*. Различающие последовательности строятся для всех триггеров также с использованием генетического алгоритма и многозначного алфавита. Следует отметить, что условие неопределенных значений для остальных триггеров является достаточно жестким, и не для всех триггеров могут быть найдены различающие входные последовательности.

Представленные характеристические последовательности генерируются с использованием генетического алгоритма нижнего уровня, где в качестве особи используются двоичные последовательности (таблицы) и генетические операторы представленные ниже.

Эволюционный алгоритм верхнего уровня

На верхнем уровне используется генетический алгоритм, представленный в [1] с некоторыми модификациями. Во-первых, в начальную популяцию (и в процессе генерации) включаются не только случайные входные последовательности, но и характеристические последовательности, построенные на нижнем уровне. Во-вторых, расширен набор генетических операторов, которые используются при генерации новых особей. В него включены следующие генетические операторы.

1. *Классический одноточечный кроссинговер*. В этом случае двоичная таблица интерпретируется одной двоичной строкой, которая получается в результате последовательной конкатенации («склеивания») строк. Поскольку в оперативной памяти таблица представляется именно в таком виде, то этот тип кроссинговера имеет простую реализацию. Отметим, что часто используемый на практике оператор горизонтального кроссинговера является частным случаем одноточечного кроссинговера.

2. *Горизонтальный кроссинговер*, где случайно выбирается момент времени k (точка скрещивания) и родительские последовательности обмениваются подпоследовательностями после строки k.

3. *Вертикальный кроссинговер,* где обмен между родительскими особями производится случайно выбранными столбцами. Напомним, что столбец соответствует входной последовательностим, подаваемой на один вход схемы.

4. *Свободный вертикальный кроссинговер,* представленный на рис.3, который выполняется следующим образом. Для каждой строки k (входного набора) родительських двоичных таблиц определяется своя точка скрещивания t_k. Затем каждая пара строк (двоичных входных наборов) обменивается подстроками после точки скрещивания t_k , что показано на рис.3.

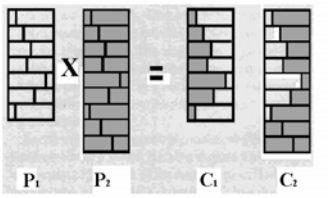

Рис. 3. Свободный вертикальный кроссинговер

5. *Однородный кроссинговер* отличается от предыдущих видов [1]. Здесь каждый ген потомка создается путем копирования соответствующего гена из первого или второго родителя, то есть каждая

позиция потенциально является точкой кроссинговера. Для этого случайным образом генерируется двоичная маска кроссинговера той же длины (с тем числом бит), что у хромосом родителей. Четность бита маски показывает родителя, из которого копируется ген потомка. Каждый бит потомка копируется из 1-го или 2-го родителя в соответствии со значением этого бита маски.

6. *Структурный кроссинговер* фактически является обобщением вертикального кроссинговера, в котором обмен между родителями производится столбцами. Здесь обмен также производится столбцами, соответствующими одной древовидной подсхеме. Предварительно схема должна быть разбита на древовидные подсхемы. Входы, которые «питают» одну древовидную подсхему, относятся к одной и той же группе. Здесь обмен производится группами столбцов, соответствующих одной и той же древовидной подсхеме. Поэтому здесь обмен производится более направленно для внутренних линий схемы, что повышает эффективность поиска тестовых последовательностей.

Итак, скрещивание реализуется в виде приведенных независимых операций скрещивания, выбор в процессе работы, между которыми, происходит с вероятностью $P_1, P_2,\ldots, P_6$, где значения P_i подбираются экспериментально и $P_1+ P_2+ P_3+ P_4+ P_5+P_6=1$.

Далее, как обычно, к полученным потомкам применяются операторы *мутации*. Здесь также производится выбор одной из возможных операций:

1. Удаление одного входного вектора из случайно выбранной позиции.

2. Добавление одного входного вектора в случайную позицию.

3. Случайная замена битов в тестовой последовательности.

Выбор между тремя операторами мутации также производится случайно с вероятностями $P_{мут1}$ и $P_{мут2}$ с распределением $0<P_{мут1}<P_{мут2}<1$.

Выводы

Предложен двухуровневый эволюционный алгоритм построения проверяющих тестов с использованием характеристических последовательностей и стратегии кратного наблюдения выходных сигналов, что позволяет повысить полноту проверяющих тестов. Перспективным является объединение иерархических и параллельных [2] эволюционных алгоритмов генерации тестов.

Литература

1. Скобцов Ю.А., Скобцов В.Ю. Логическое моделирование и тестирование цифровых устройств. – Донецк: ИПММ НАНУ, 2005. – 436 с.

2. Скобцов Ю.А., Иванов Д.Е., Эль-Хатиб А.И. Распределенные генетические алгоритмы генерации проверяющих тестов цифровых систем. // Радіоелектронні і комп’ютерні системи. – 2007. – № 7. – С.176-181.

Поступила в редакцию 21.01.2008

Рецензент: д-р техн. наук, проф. А.Ю. Соколов, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.