УДК 519.713:681.3

Е.В. УКОЛОВА

Саратовский государственный университет им. Н.Г. Чернышевского, Россия

О ПОСТРОЕНИИ КОНТРОЛИРУЮЩЕЙ ТЕСТОВОЙ ПОСЛЕДОВАТЕЛЬНОСТИ С ПРИМЕНЕНИЕМ ГЕНЕТИЧЕСКОГО АЛГОРИТМА

Предложен метод построения контролирующих тестов для дискретных устройств, основанный на генетическом алгоритме. Работоспособность и эффективность метода проверялись путем построения тестов для схем, приведенных в каталоге ISCAS'89. При построении тестов рассматривались одиночные константные неисправности. Неисправности эмулируются программной установкой соответствующего объекта в памяти в особое состояние, при котором он выполняет функцию неисправного элемента. Генерация тестов осуществлялась при различных значениях и конфигурациях параметров генетического алгоритма: изменялись вид селекции, мощность популяции, вероятность мутации, процент элитных особей, максимальное количество итераций. Для повышения эффективности работы генетического алгоритма в программу была включена эвристическая функция подбора оптимальной доли единиц во входной последовательности. Результаты экспериментов показали, что синтез тестов осуществляется за приемлемое время.

контролирующий тест, генетический алгоритм, генерация тестов

Введение

Развитие технологий производства приводит не только к улучшению различных характеристик дискретных устройств, увеличению количества выполняемых ими функций, но также к усложнению их элементной базы. Как следствие, растет вероятность появления неисправностей и сложность их обнаружения. Таким образом, весьма актуальной задачей является разработка методов обнаружения неисправностей. Один из таких методов заключается в подаче на устройство теста – специальной последовательности, особенность которой заключается в том, что реакции на нее исправного и неисправного устройств различаются. В качестве входных используются либо рабочие воздействия (т.е. такие, которые поступают на устройство в процессе его функционирования по назначению), либо специально генерируемые тестовые воздействия.

Учитывая сложность современных дискретных устройств, возникает необходимость автоматизации процесса построения тестов. Одним из способов добиться этого является применение генетического алгоритма (ГА).

Описание модели

Для автоматизации процесса построения тестов необходимо программно моделировать дискретные устройства (ДУ). Рассматривают несколько уровней и областей проектирования [1]. Области делятся на поведенческую, структурную и физическую. Для каждой из них рассматривают схемный, логический, системный уровни, а также уровень языков регистровых передач.

В данной работе рассматривается моделирование на логическом уровне. Логическое моделирование включает в себя построение математической модели дискретного устройства, а также анализ поведения построенной модели на заданной последовательности входов.

При этом элементы дискретного устройства описываются с помощью булевых функций, которые отражают их функционирование. Построение тестов осуществляется для последовательностных дискретных устройств (с памятью). Последовательностные дискретные устройства – это такие устройства, у которых значения их выходных сигналов в текущий момент времени зависит от значений входных сиг-

налов в текущий момент времени, а также от внутреннего состояния устройства в предыдущий момент времени. Под внутренним состоянием дискретного устройства понимается совокупность состояний 0 или 1 триггеров [2].

Для моделирования необходимо наличие структурной модели дискретного устройства, которая отражает не только логику функционирования, но также связи между компонентами и внешней средой.

Такая модель представляется ориентированным графом с логическими элементами, входами, выходами и узлами разветвления в качестве вершин. Дугами являются соединения логической схемы. Логическая схема называется правильной, если выходы никаких двух элементов не соединены вместе, и каждая из функций, реализуемых на выходах дискретного устройства, может быть представлена как функция выхода конечного автомата. Логические элементы могут быть двух типов:

– элементы, функционирование которых описывается булевыми функциями;

– элементы памяти.

В работе структурная модель строилась на основе информации, извлекаемой из файлов международного каталога ISCAS'89. Здесь каждая схема задается входами, выходами, компонентами и связями между ними.

Для корректного моделирования схемы необходимо распределить вершины по уровням. В последовательностной схеме для вычисления уровней элементов сначала производится обрыв линий обратных связей, и полученным псевдовходам присваивается уровень 0. Нулевой уровень присваивается также внешним входам. Уровень l_i элемента e_i, на входы которого поступают сигналы с выходов элементов $e_1,\ldots,e_p$, уровни которых соответственно равны $l_1,\ldots,l_p$, определяется таким образом:

$$l_i = 1 + \max_{1\le j\prec p}(\ l_j\ ).$$

Программная реализация

Функционирование дискретных устройств моделировалось в двоичном алфавите, что достаточно точно отражает поведение ДУ в статике, при установившихся значениях, хотя при этом не учитываются переходные процессы, возникающие при смене входных сигналов и обусловленные временными характеристиками элементов. Предполагается, что триггеры всех схем к началу моделирования сброшены в 0.

В разработанной программе синтеза тестов описание схемы транслируется в программную модель схемы. На каждом такте входной вектор проходит по уровням от первичных входов к первичным выходам. Неисправности эмулируются программной установкой соответствующего объекта в памяти в особое состояние, при котором он выполняет функцию неисправного элемента. При построении теста входная последовательность подается на исправную схему, а затем на схемы, соответствующие неисправностям, описанным в соответствующих файлах каталога ISCAS'89 (одиночным константным). Используя полученные выходные сигналы, делается вывод о том, насколько хорошей является поданная входная последовательность, т.е. насколько высоко ее качество как теста. Качество тестов можно оценивать по разным признакам: полнота, длина и др. Алгоритм генерации тестов также оценивается по различным критериям, таким как быстродействие, объем требуемой памяти, адекватность получаемых результатов и т. д.

Программа, с помощью которой производится генерация теста, написана на C++, Visual Studio 2005. Скорость моделирования зависит от размера схемы. Ниже приведены примеры для некоторых из схем.

1. Для схемы s27 (три триггера, восемь логических элементов) понадобилась меньше 15 секунд для того, чтобы прошло 100 итераций генетического алгоритма, на каждой из которых происходит моде-

лирование для исправного и 32 неисправных дискретных устройств для каждой из 25 особей в популяции.

2. Для схемы s713 (19 триггеров, 139 логических элементов) при тех же параметрах генетического алгоритма (рассматривается 581 неисправность) понадобилось около 50 минут, из которых 99 итераций заняли чуть больше получаса, а самая первая – остальное время. Такая разница во времени работы первой и остальных итераций обусловлена тем, что изначально длины векторов, при помощи которых происходит хранение данных, принимаются очень большими по сравнению с последующими.

Для того чтобы увеличить эффективность построения тестов, перед началом работы генетического алгоритма запускается случайный поиск, в процессе которого определяется частота появления во входной последовательности единиц, при которой, вероятно, будет получен достаточно короткий тест.

Частота определяется простым эвристическим подбором. Для всех частот от 0 до 100 с шагом 10 строится случайная последовательность, при этом определяется, в каком случае получается наиболее короткий тест. Далее такой же поиск происходит в некоторой небольшой окрестности найденной ранее предполагаемой частоты.

Если все попытки найти такую «хорошую» частоту оказываются неудачными, т.е. не удается построить тест, обеспечивающий заданную полноту покрытия, то частота единиц принимается равной 0,5.

Для ускорения работы программы используется как можно меньший объем памяти, т.е. в памяти хранится только текущее состояние схемы, либо исправной, либо соответствующей некоторой неисправности.

Генетический алгоритм

В настоящее время достаточно сильно развиваются методы эволюционных вычислений, к которым относятся и генетические алгоритмы. Генетический алгоритм представляет собой адаптивный поисковый метод [3], который основан на селекции лучших элементов в популяции, подобно эволюционной теории Ч. Дарвина. Генетический алгоритм эффективно использует информацию, накопленную в процессе эволюции. Также в генетический алгоритме используется кроссовер, при помощи которого генерируются новые особи путем скрещивания уже существующих особей, и мутация, при помощи которой каким-либо способом резко изменяются свойства потомка.

В простом генетическом алгоритме присутствуют, таким образом, три основных оператора: репродукция, кроссовер и мутация.

Все эти операторы имеют различные реализации и виды. В программе использованы турнирная и рулеточная виды селекции [4], а также два вида кроссовера, описанные ниже.

Возможна также вариация других параметров: вероятности мутации, процента элитных особей, требуемой полноты покрытия, максимальное количество итераций и число особей в популяции.

При решении практических задач с использованием эволюционных методов необходимо выполнить следующие четыре этапа.

1. Выбрать способ представления решения.
2. Разработать операторы случайных изменений.
3. Определить законы выживания решения.
4. Создать начальную популяцию.

Каждому допустимому решению *и* сопоставляется вектор битовых строк, составляющий хромосому. Изначально все хромосомы в популяции имеют некоторую достаточно большую длину. Но уже после первой итерации, когда будет подсчитана фитнес-функция для каждой особи (хромосомы), их длины могут заметно уменьшиться. Таким образом, в следующей популяции будут представлены хромосомы с различными длинами.

При подсчете фитнес-функции каждой хромосомы учитывается заданный заранее параметр – пол-

нота покрытия. В этом случае значение фитнес-функции принимается равным номеру наименьшего шага, при котором необходимое количество неисправных модификаций выдали реакции, отличные от реакции исправной схемы. В противном случае, значение фитнес-функции принимается равным бесконечности.

Таким образом, длина входного вектора становится равным значению фитнес-функции, если это значение меньше бесконечности, и остается тем же в противном случае.

После подсчета фитнес-функций хромосом происходит сортировка популяции по возрастанию значений фитнес-функций, и далее производится отбор родителей для получения следующего поколения. Родители отбираются либо с помощью рулеточной, либо с помощью турнирной селекции – этот параметр задается извне. Число родителей равно мощности популяции.

В новую популяцию попадает заданное количество элитных особей, т.е. таких, которые без изменений переносятся из существующей популяции. Далее новая популяция пополняется до необходимого количества путем выполнения оператора кроссовера, применяемого к родителям. С равной вероятностью для каждой пары родителей выбирается один из двух видов кроссовера:

1. Кроссовер, при котором деление хромосом происходит «перпендикулярно входам». В этом случае находим точки деления первого и второго родителей – m_1 и m_2 соответственно, где $0 \le m_i \le L_i$, $i = \overline{1,2}$ и L_i – длина i-го родителя. К потомку отходит m_1 первых входных воздействий первого родителя и m_2 последних входных воздействий второго родителя. Таким образом, длина входной последовательности, задаваемой потомком, равна $m_1 + m_2$.

2. Одноточечный кроссовер, при котором деление хромосом происходит «параллельно входам». Здесь случайным образом выбирается точка деления m, $0 \le m \le n$, где n – число первичных входов дискретного устройства. Далее к потомку отходят все входные воздействия первого родителя по первым m входам, а также входные воздействия второго родителя по оставшимся $n - m$ входам. Причем, если длина входной последовательности одного из родителей больше, то недостающие биты заполняются случайным образом.

Данная процедура выполняется столько раз, какова мощность популяции за вычетом числа элитных особей. К каждому потомку далее применяется оператор мутации с заданной заранее вероятностью.

После этого формирование новой популяции считается завершенным.

Работа генетического алгоритма завершается после выполнения заданного числа итераций.

Результаты экспериментов

Целью данной работы было проанализировать, при каких настройках параметров генетического алгоритма построение тестов происходит наиболее эффективно, т.е. за наименьшее время и оптимальной длины для заданного покрытия.

Большую роль в генетических алгоритмах играют случайные изменения. Это происходит даже в большей степени не за счет мутации, а при кроссовере, так как построение новой входной последовательности путем кроссовера изменяет реакции схемы практически непредсказуемо. Некоторую стабильность придают элитные особи, не позволяющие исчезнуть достаточно хорошим признакам.

Были протестированы несколько схем. Можно утверждать, что генетический алгоритм более эффективный метод поиска, чем просто случайный подбор.

Неодинаково влияние изменений параметров генетического алгоритма для различных схем. Например, для схем s27, s510 практически одинаковые результаты получились как при использовании ру-

леточной, так и при использовании турнирной селекции. Но для большинства остальных схем при рулеточной селекции длины тестов в среднем получались более короткими. При этом полнота покрытия при построении тестов бралась равной от 70 до 100%, количество особей – 20 или 25 (увеличение этого числа до 40 не привело к улучшению результата, а лишь ухудшилось время работы программы), максимальное число итераций – 100, вероятность мутации – 1%, число элитных особей – 10% от мощности популяции. Изменения вероятности мутации до 15% не оказывало какого-то заметного эффекта.

Таблица 1

Результаты численных экспериментов
(roul – рулеточный тип селекции, tour – турнирный)

Название схемы	Полнота покрытия, (%)	Тип селекции	Средняя длина теста
S27	100	roul	6,32
S27	100	tour	6,57
S298	85	roul	49
S298	85	tour	61,25
S298	80	roul	26,25
S298	80	tour	30,1
S298	70	roul	18,42
S298	70	tour	19,68
S2081	100	roul	801
S2081	100	tour	1204,5
S2081	95	roul	381,38
S2081	95	tour	412
S2081	90	roul	220
S2081	90	tour	249,8
S713	80	roul	112,5
S713	80	tour	116
S510	100	roul	188,67
S510	100	tour	215,33
S510	80	roul	29
S510	80	tour	29

Такие различные значения длин тестов получаются из-за неодинаковой сложности схем.

Например, в схеме s27 содержится всего 3 триггера, 8 логических элементов и 2 инвертора, в схеме s298 – 14 триггеров, 44 инвертора и 75 логических элементов. Схема s2018 содержит 8 триггеров, 38 инверторов и 66 логических элементов, s510 – 6 триггеров, 32 инвертора, 179 логиеческих элементов.

Самая большая из рассмотренных схем – s713 – содержит 19 триггеров, 254 инвертора и 139 логических элементов. Чем сложнее схема, тем дольше осуществляется построение схеы, и тем более длинным получается тест.

Литература

1. Скобцов Ю.А., Скобцов В.Ю. Логическое моделирование и тестирование цифровых устройств. – Донецк: ИПММ НАН Украины, ДонТУ. – 2005. – 344 с.

2. Барашко А.С., Скобцов Ю.А., Сперанский Д.В. Моделирование и тестирование дискретных устройств. – К.: Наук. думка, 1992. – 282 с.

3. Goldberg, D.E. Genetic Algorithms in Search, Optimization, and Machine Learning. – Addison-Wesley, Reading, Mass., 1989. – 620 p.

4. Tobias Blickle, Lothar Thiele. A Comparison of Selection Schemes used in Genetic Algorithms. Computer Engineering and Communication Networks Lab (TIK) Swiss Federal Institute of Technology (ETH).

Поступила в редакцию 27.02.2007

Рецензент: д-р техн. наук, проф. В.М. Илюшко, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.