Анализ экспериментальных данных, полученных для приведенных выше и других схем, и их сравнение с результатами, опубликованными в [4], говорит о достаточно высокой эффективности предложенного алгоритма синтеза случайных тестов.

Основным достоинством приведенного алгоритма является необходимость моделирования только исправного ДУ при достижении высокого уровня покрытия тестируемых схем тестовыми наборами. Моделирование с неисправностями используется исключительно для получения диагностической информации о построенном тесте.

Библиографический список

1. *Agrawal V.D.* An Information Theoretic Approach to Digital Fault Testing // IEEE Transactions on Computers. 1981. V. 30. P. 582–587.
2. *Holland J.H.* Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, 1975.
3. *Суворова Е.А., Шейнин Ю.Е.* Проектирование цифровых систем на VHDL. СПб.: БХВ-Петербург, 2003.
4. *Скобцов Ю.А., Скобцов В.Ю.* Логическое моделирование и тестирование цифровых устройств. Донецк: Изд-во Донецк. техн. ун-та, 2005.

УДК 519.713:681.3

СИНТЕЗ КОНТРОЛИРУЮЩИХ ТЕСТОВЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ С ПРИМЕНЕНИЕМ ГЕНЕТИЧЕСКОГО АЛГОРИТМА

Е.В. Уколова

Саратовский государственный университет,
кафедра математической кибернетики и компьютерных наук
E-mail: b4456@yandex.ru

В статье 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. Для моделирования работы схем и генерации тестов была написана программа на C++, Visual Studio 2005. 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. В статье приведены результаты построения тестов, а также сравнение с данными других авторов.

The Construction of Supervisory Test with the Use of the Genetic Algorithm

E.V. Ukolova

The article describes a control test generation method for discrete devices based on the genetic algorithm. The method operability and effectiveness have been checked by means of creating tests for circuits listed in the ISCAS' 89 catalogue. The C++(Visual Studio 2005) program has been implemented in order to simulate circuit and generate tests. Faults are simulated by programmatically setting an appropriate object in storage to a special state, in which it acts as a faulty component. Test generation has been executed using different values and configurations of the genetic algorithm parameters: selection, population capacity, mutation probability, percent of elite individuals, maximal amount of iterations were changed. The final results and comparison with results of other authors are included into the report.

ВВЕДЕНИЕ

В современном мире бурно развиваются технологии производства, функции дискретных устройств становятся все более разнообразными, а их технические характеристики непрерывно улучшаются. В связи с этим происходит постоянное усложнение элементной базы дискретных устройств. 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 (то есть такие, которые поступают на устройство в процессе его функционирования по назначению), либо специально генерируемые тестовые воздействия. Сложность современных дискретных устройств приводит к необходимости автоматизации процесса построения тестов. Одним из способов достижения этого является применение генетического алгоритма (ГА).

1. 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 трехзначных 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 сигналов в текущий момент времени, а также от внутреннего состояния устройства в предыдущий момент времени. Под внутренним состоянием дискретного устройства понимается совокупность состояний 0, 1 или u 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:

1) элементы, функционирование которых описывается трехзначными функциями;

2) элементы памяти.

В статье 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: $l_i = 1 + \max\limits_{1 \le j \le p}(l_j)$.

2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

Функционирование дискретных устройств моделировалось в троичном алфавите, что достаточно точно отражает поведение ДУ. Предполагается, что все триггеры ДУ перед началом моделирования находятся в неопределенном состоянии (u).

В разработанной программе синтеза тестов описание схемы транслируется в программную модель схемы. На каждом такте входной вектор проходит последовательно 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, в которые внесены неисправности, описанные в соответствующих файлах каталога ISCAS'89 (одиночные константные). Используя полученные выходные сигналы, производится анализ того, насколько «хорошей» является поданная входная последовательность, то есть 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 данные о быстродействии программы для некоторых из схем.

1. Для схемы s27 (три триггера, восемь логических элементов) понадобилось меньше 24 секунд для того, чтобы прошло 100 итераций генетического алгоритма, на каждой из которых происходит моделирование для исправного и 32 неисправных дискретных устройств для каждой из 20 особей в популяции.

2. Для схемы s713 (19 триггеров, 139 логических элементов, рассматривается 581 неисправность), где осуществляется всего 50 итераций с мощностью популяции, равной 15 особям, понадобилось около 150 минут. Такая разница во времени работы обусловлена тем, что длины векторов, при помощи которых происходит хранение данных, намного больше, чем для схемы s27.

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. Правда, при использовании троичного алфавита данный эвристический метод подбора частоты работает очень медленно и не всегда его применение оправдано.

Частота определяется простым эвристическим поиском. Для всех частот от 0 до 1 с шагом 0,1 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, то есть 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, то есть в памяти хранится только текущее состояние схемы, либо исправной, либо соответствующей некоторой неисправности.

3. ГЕНЕТИЧЕСКИЙ АЛГОРИТМ

В настоящее время достаточно бурно развиваются методы эволюционных вычислений, к которым относятся и генетические алгоритмы. Генетический алгоритм (ГА) представляет собой адаптивный поисковый метод [3], который основан на селекции лучших особей в популяции, подобно эволюционной теории Ч. Дарвина. Генетический алгоритм эффективно использует информацию, накопленную в процессе эволюции. В генетическом алгоритме также используется кроссовер, при помощи которого генерируются новые особи путем скрещивания уже существующих особей, и мутация, при помощи которой могут быть резко изменены свойства потомка.

В простом генетическом алгоритме присутствуют, таким образом, три основных оператора: репродукции, кроссовера и мутации.

Все эти операторы имеют различные реализации и виды. В нашей 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, максимального количества итераций и числа 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, в частности, кроссовер и мутацию;

3) определить законы выживания решения;

4) создать начальную популяцию.

Каждому допустимому решению T сопоставляется вектор битовых строк, составляющий хромосому. Таким образом, хромосома задает входную последовательность, подаваемую на дискретное устройство. Если входная последовательность (хромосома) будет обладать таким свойством, что исправное

устройство и определенное количество неисправных модификаций устройства будут выдавать различные реакции на нее, то эту последовательность назовем тестом.

При подсчете фитнес-функции каждой хромосомы учитывается заданный заранее параметр — полнота покрытия. В этом случае значение фитнес-функции принимается равным номеру наименьшего шага, при котором количество неисправных модификаций, выдавших реакции, отличные от реакции исправной схемы, таково, что достигнута заданная полнота покрытия. В противном случае значение фитнес-функции принимается равным бесконечности.

Изначально все хромосомы в популяции имеют некоторую достаточно большую длину. Но уже после первой итерации, когда будет подсчитана фитнес-функция для каждой хромосомы, их длины могут заметно уменьшиться. Это происходит вследствие того, что если при подаче очередного входного сигнала количество неисправностей, которые обнаружились, обеспечивает заданную полноту покрытия, то оставшиеся входные сигналы из хромосомы удаляются. Вместе с тем в процессе работы ГА возможно и увеличение длин входных последовательностей, например, в результате применения кроссовера. Поэтому в следующей популяции будут представлены хромосомы с различными длинами.

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 извне количество элитных особей, то есть таких, которые без изменений переносятся из существующей популяции в следующую. Далее, применяя кроссовер к выбранным случайным образом парам родителей, получаем особей, которых помещаем в новую популяцию. Это происходит до тех пор, пока количество особей в этой популяции не достигнет числа, равного мощности популяции. Для каждой выбранной пары родителей реализуется один из двух типов 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 $0 \leq m_i \leq L_i, i = \overline{1,2}$, и L_i — длина i-ого 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, $0 \leq m \leq n$, 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.

Тип кроссовера определяется параметром извне.

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.

4. РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТОВ

Целью данной работы было проанализировать, при каких значениях параметров генетического алгоритма построение тестов происходит наиболее эффективно (наименьшее время и оптимальная длина для заданного покрытия).

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

Для того чтобы не потерять лучшие из найденных на данный момент времени решения, в процессе работы генетического алгоритма отбираются элитные особи.

Отметим, что тип кроссовера в проведенных экспериментах не оказывал заметного влияния, если моделирование осуществлялось в двоичном алфавите. Но при использовании троичного алфавита выбор различных типов селекции и кроссовера приводил к изменению средней длины получаемых тестов.

Кроме того, влияние изменений параметров генетического алгоритма для различных схем оказалось неодинаковым. Для многих схем наиболее короткие тесты генерировались, когда использовался турнирный тип селекции и кроссовер «перпендикулярно входам» (тип 1). В некоторых случаях после окончания работы ГА так и не удавалось получить входную последовательность, которая обнаруживала бы заданное количество неисправностей. При использовании кроссовера «параллельно входам» такая ситуация возникала более редко. Достаточно хорошие результаты были получены при использовании обоих типов кроссовера одновременно. Для некоторых схем, в частности для схемы s27, наилучший результат был получен при использовании рулеточной селекции.

При использовании одного и того же типа селекции более короткие тесты для большинства схем генерировались при использовании кроссовера типа 1. При этом полнота покрытия при построении тестов задавалась в диапазоне от 70 до 100%, количество особей — 15 или 25 (увеличение этого числа до 40 не привело к улучшению результата, а лишь увеличилось время работы программы), максимальное число итераций — 100, вероятность мутации — 1%, число элитных особей — 10% от мощности популяции. Результаты некоторых экспериментов приведены в таблице.

Результаты численных экспериментов

Название схемы	Полнота покрытия	Тип селекции/ тип кроссовера	Средняя длина теста
S27	100	roul/1	8.90
S27	100	tour/1	9.66
S27	100	roul/2	19.50
S27	100	tour/2	19.56
S298	86	roul/1	171.25
S298	86	tour/1	130.25
S298	86	roul/2	465.00
S298	86	tour/2	239.00
S713	81,5	roul/1	779.67
S713	81,5	tour/1	465.67
S713	81,5	roul/2	1081.50
S713	81,5	tour/2	1434.33
S386	70	roul/1	1172
S386	70	tour/1	885
S386	70	roul/2	2978.4
S386	70	tour/2	2262.33

Примечание. roul — рулеточный тип селекции, tour — турнирный.

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. Самая большая из рассмотренных схем — s713 — содержит 19 триггеров, 254 инвертора и 139 логических эелементов. Чем сложнее схема, тем дольше осуществляется моделирование схемы и обычно генерируется тест большей длины.

Отметим, что в процессе работы были получены тесты, которые могут обнаружить больше неисправностей, чем представлено в [1], причем длины тестов часто короче приведенных там же. Например, для схемы s298 построенный нами тест обнаруживает 265 неисправностей (против 254 в [1]), для схемы s713 наш тест обнаруживает 476 неисправностей (против 469 в [1]), для схемы 386 наш тест обнаруживает 269 – 271 неисправность (против 263 в [1]).

Библиографический список

1. *Скобцов Ю.А., Скобцов В.Ю.* Логическое моделирование и тестирование цифровых устройств. Донецк: Изд-во Донецк. техн. ун-та, 2005.

2. *Барашко А.С., Скобцов Ю.А., Сперанский Д.В.* Моделирование и тестирование дискретных устройств. Киев: Наук. думка, 1992.

3. *Goldberg D.E.* Genetic Algorithms in Search, Optimization, and Machine Learning. Reading: Addison-Wesley, 1989.

4. *Blickle T., Thiele L.* A Comparison of Selection Schemes used in Genetic Algorithms // TIK - Report. № 11, December. Zurich: ETH, 1995. P.65