УДК 681.518

Д.Е. Иванов, Ю.А. Скобцов
Институт прикладной математики и механики НАН Украины

Ускорение работы генетических алгоритмов при построении тестов

Рассматривается проблема построения тестов для синхронных последовательностных схем с помощью генетического алгоритма. Основанная на моделировании генетическая стратегия построения тестов показывает сравнимые с детерминированными методами результаты, обладая рядом преимуществ: прозрачность стратегии и простота реализации. Предлагаются методы, позволяющие повысить быстродействие генетического алгоритма: оптимизированный способ построения новой популяции и параллельное вычисление оценочных функций.

Введение

Проблема построения тестов для последовательностных устройств является одной из самых сложных в технической диагностике. В последнее время разработан ряд методов построения тестовых наборов, позволяющих добиться высокой полноты тестов как для комбинационных, так и для последовательностных схем. Все они могут быть условно разделены на три группы. К первой относятся структурные методы, являющиеся расширением для последовательностных схем метода ветвей и границ [1]. Результаты такого подхода зависят от эвристик, используемых при направлении поиска. К существенным недостаткам метода относятся сложность реализации и сложность применения к большим схемам. Во вторую группу включаются методы, основанные на символьном моделировании [2]. Их суть заключается в получении и обработке булевых функций, реализующих схему. Методы символьного моделирования (так же, как и методы первой группы) являются алгоритмами и показывают хорошие результаты для небольших и средних схем. Однако они практически полностью неприменимы для схем, содержащих более двух десятков триггеров. Третья группа методов основана на псевдослучайной генерации тестовых наборов и дальнейшей их модификации с учетом моделирования (исправного или с неисправностями) [3]. Распространение этой группы методов обусловлено появлением быстрых программ моделирования с неисправностями [4].

Предложенный Голдбергом в генетический подход хорошо зарекомендовал себя при решении задач переборного типа с NP полным решением [5]. Используемый в данном подходе генетический алгоритм относится к третьей группе методов построения тестов. Показывая сопоставимые с детерминированными методами результаты, он является гораздо более простым по структуре и в реализации. Цель статьи заклю-

чается в разработке практических методов, реализованных в алгоритме построения тестовых наборов для последовательностных схем, которые позволяют существенно повысить скорость работы генетического алгоритма. Статья имеет следующую структуру. Во втором разделе кратко описана адаптация генетического алгоритма к проблеме построения тестов последовательностных схем. В третьем разделе представлены программы моделирования, используемые в алгоритме. В четвертом разделе описывается способ построения новой популяции, позволяющий избежать повторного вычисления оценочной функции для особей, вошедших в нее без изменения. В заключительной части приведены результаты машинных экспериментов, проведенных с программной реализацией алгоритма на контрольных схемах из международного каталога ISCAS-89.

Генетический алгоритм

Для решения проблемы генерации тестов в последнее время получили распространение генетические алгоритмы [6]. Применительно к проблеме генерации тестов суть генетического алгоритма заключается в следующем. В начальный момент времени (обычно, случайным образом) генерируется ряд тестовых последовательностей – популяция (рис.1).

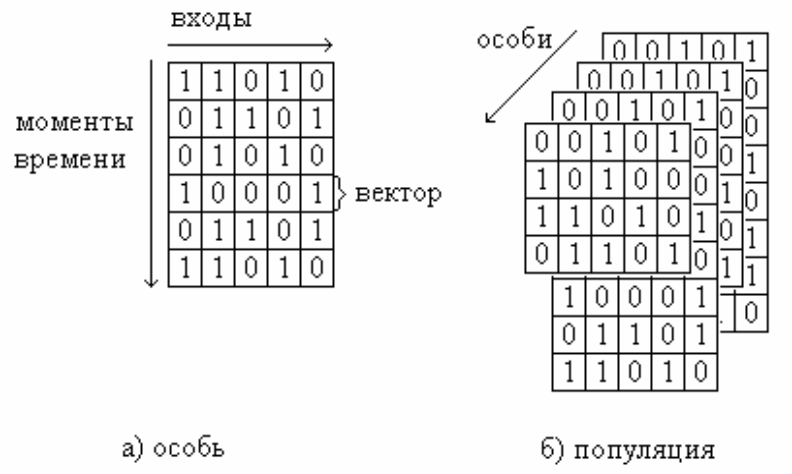

Рис. 1. Кодирование особей и популяций.

Каждая последовательность называется особью и является частичным решением проблемы. Далее на основании некоторых критериев вычисляется оценочная функция каждой особи (тестовой последовательности), характеризующая относительную способность решения поставленной проблемы. Обычно для вычисления оценочной функции используется моделирование с неисправностями, реже – исправное моделирование. Из текущего поколения особей строится следующее с целью либо найти решение, либо улучшить оценку каждой особи. Цель данной операции – получение особей в популяции, обладающих новыми свойствами. Новое поколение строится следующим образом. Из популяции случайным образом выбираются две особи.

Вероятность выбора особи полагается пропорциональной оценочной функции. Копии выбранных особей сразу помещаются в новую популяцию. После этого к данным особям применяются модифицирующие операции. Обычно используют два вида таких операций: скрещивание и мутацию. Скрещивание производится в одной или нескольких точках (рис.2).

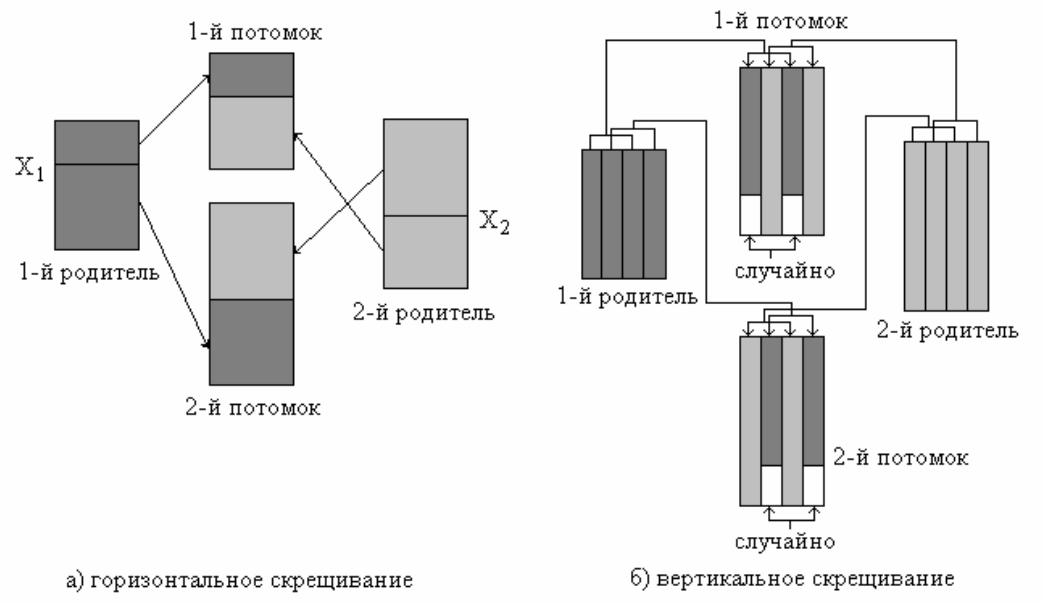

Рис.2. Операция скрещивания.

Далее с некоторой вероятностью в двух новых особях происходит мутация битов: ноль изменяется на 1 и наоборот, после чего модифицированные особи помещаются в новую популяцию. Так происходит до тех пор, пока в новой популяции не наберется необходимое число особей. Порождение новых популяций прекращается, когда найдено удовлетворительное решение, либо произведено вычисление заранее заданного числа поколений.

Таким образом, чтобы задать генетический алгоритм, необходимо определить понятия особи, популяции, операции скрещивания и мутации, задать оценочную функцию. Очевидно также, что эффективность генетического алгоритма зависит от целого ряда параметров: размера популяции, метода выбора особей из предыдущей популяции, скрещивания и мутации, а также вида оценочной функции.

На основе описанного выше подхода авторами был разработан и реализован алгоритм генерации тестов для последовательностных схем. Структурно он состоит из трех фаз. В фазе 1 из списка непроверенных неисправностей необходимо выбрать одну целевую неисправность, которая может быть активизирована, т.е. значения сигналов в исправной и неисправной схемах распространяются на внешние псевдовыходы. Для достижения этой цели используется псевдослучайная генерация последовательностей и дальнейшее моделирование с неисправностями на данных последовательностях. В фазе 1 используется программа моделирования неисправных схем, описанная в [7]. Основной в данном алгоритме является

фаза 2, цель которой с помощью генетического алгоритма улучшить активизирующую последовательность, полученную в фазе 1, чтобы она стала проверяющей для целевой неисправности. Фаза 3 используется для удаления из списка тех неисправностей, которые обнаруживаются построенной в фазе 2 последовательностью. Общая структура алгоритма в виде псевдокода представлена ниже.

```
генерация_тестов (схема)
{
   while(не достигнута заданная полнота)
   {
      цель=активизировать_неисправность(); // Фаза 1
      if( цель == НЕТ_НЕИСПРАВНОСТИ )
       goto конец;
      последовательность=GA:_генерация_тестовой_последовательности
      (цель) //Фаза 2
      if( последовательность != НЕТ_ПОСЛЕДОВАТЕЛЬНОСТИ )
         моделирование_с_неисправностями( последовательность ); // Фаза 3
      else // не смогли найти тестовую последовательность для целевой
         неисправности отметить_неисправность_как_непроверяемую();
   }       // конец while – достигнута заданная полнота
   конец:
}          // конец алгоритма
```

Программы моделирования и вычисление оценочных функций

Для повышения быстродействия алгоритма построения тестов в программную реализацию интегрированы две программы параллельного моделирования с неисправностями. Первая программа [7] используется в фазе 1 для проверки активизации произвольной неисправности случайно сгенерированной последовательностью. На данном этапе мы рассматриваем только потенциальных кандидатов на тестовую последовательность, поэтому удалять неисправности, которые они обнаруживают, из списка неисправностей не нужно. Это достигается введением в программу моделирования «пробного» режима. После моделирования в таком режиме мы будем знать, сколько неисправностей проверилось данной тестовой последовательностью из заданного списка неисправностей, но сам список не изменится. В списках неисправных элементов состояний [7] хранятся значения сигналов псевдовыходов неисправной схемы. Поэтому, если данный список для произвольной неисправности не пуст, значит ее влияние распространилось на внешние псевдовыходы, или, другими словами, произошла ее активизация. «Пробный» режим моделирования с неисправностями применяется в фазе 1 алгоритма. При этом моделирование может

выполняться либо до активизации первой неисправности из списка еще непроверенных, либо на всех особях-кандидатах. В первом случае в качестве целевой выбирается активизированная неисправность. Во втором случае, если активизировано несколько неисправностей, возможно применение некоторых критериев. Например, в качестве целевой выбирается та неисправность, в присутствии которой в схеме наблюдается наибольшая активность, либо влияние неисправности которой распространилось на большее число псевдовыходов, либо применить комбинацию данных критериев. Возможно также адаптировать для этой цели генетический алгоритм, что позволит выбирать самую активную неисправность. При этом оценочные функции будут аналогичны используемым в фазе 2 (см. ниже).

Моделирование с выключенным пробным режимом используется в фазе 3 алгоритма. Входом этой фазы является тестовая последовательность, которая проверяет текущую неисправность-цель. Поэтому она точно войдет в итоговый тест, и при моделировании нужно удалить из списка все неисправности, которые она проверяет. Таким образом, предотвращается их выбор в качестве целевой неисправности на следующей итерации алгоритма и точно оценивается на текущий момент полнота теста.

Самым ресурсоемким этапом генетического алгоритма, определяющим время его работы, является вычисление оценочных функций. В нашем алгоритме качество тестовой последовательности оценивается как мера отличия значений сигналов в исправной и неисправной схемах. Данная оценка основывается на предположении, что чем выше активность в неисправной схеме производит неисправность, тем легче она может быть обнаружена. Для вычисления оценочной функции используется вторая программа моделирования с неисправностями, реализующая алгоритм «параллельного моделирования по тестовым наборам» и написанная специально для работы с генетическим алгоритмом построения тестов. Суть данного алгоритма заключается в следующем. Во всех битах машинного слова моделируется одна и та же неисправность: целевая неисправность фазы 2 алгоритма, но на каждый разряд подается своя тестовая последовательность-особь. Таким образом, реализуется параллельное по особям популяции моделирование, а каждый разряд машинного слова используется для вычисления оценочной функции отдельной особи, что позволяет за один проход произвести вычисление оценочной функции сразу для всех особей в популяции. Происходит это следующим образом. После моделирования очередного тестового вектора происходит вычисление оценочной функции текущего вектора по формуле:

$$h(v, f) = f_1(v, f) + c_1 * f_2(v, f), \qquad (1)$$

где v – текущий входной набор, c_1 – константа нормирования, равная отношению числа вентилей схемы к числу триггеров; $f_1(v, f)$ и $f_2(v, f)$ – эвристические функции, определяющие:

- $f_1(v, f)$ – взвешенное число линий с различными значениями сигналов в исправной и неисправной схемах;

- $f_2(v, f)$ – взвешенное число триггеров с различными значениями сигналов в исправной и неисправной схемах.

В качестве веса выбирается мера наблюдаемости элемента схемы, вычисляемая на этапе предварительной обработки схемы.

Если текущий тестовый вектор обнаружил целевую неисправность, то его позиция показывает эффективную длину последовательности, а ее остаток отбрасывается. После того как для последовательности, моделируемой в определенном разряде, достигнута эффективная длина или произведено моделирование на последнем наборе, вычисляется оценочная функция всей последовательности:

$$H(s, f) = \sum_{i=1}^{i=длина} LH^i * h(v_i, f), \tag{2}$$

где s – анализируемая последовательность; v_i – вектор из рассматриваемой последовательности, i – позиция вектора в последовательности, f – заданная неисправность, LH – предварительно заданная константа в диапазоне $0 < LH \le 1$, благодаря которой предпочтение отдается более коротким последовательностям. Заметим, что из-за разности длин особей в популяции моделирование коротких последовательностей закончится раньше остальных. Если это произошло из-за достижения эффективной длины последовательности (обнаружена неисправность-цель), то данная последовательность сразу выбирается в качестве тестовой, поскольку обладает наименьшей длиной, а моделирование остальных прерывается.

Применение данного алгоритма существенно повышает скорость вычисления оценочных функций особей в популяции, а следовательно, и скорость работы алгоритма в целом. Недостатком такого подхода является ограниченное число особей в популяции: не более 32, что обусловлено разрядностью инструментальной ЭВМ. Однако это ограничение несущественно, поскольку число особей в популяции (MAX_ОСОБЕЙ) обычно выбирается гораздо меньше: 16 или 24 особи.

Построение новой популяции

Рассмотрим теперь подробнее операции порождения нового поколения, поскольку их оптимизация также приводит к существенному ускорению работы всего алгоритма.

Порождение нового поколения особей производится на основании оценочных функций старого поколения. Из него случайным образом пропорционально оценочным функциям выбираются две особи, чьи копии помещаются в промежуточную популяцию. К выбранным особям применяются операции скрещивания и мутации, после чего они также добавляются в промежуточную популяцию. После порождения определенного числа новых особей вычисляются оценочные функции всех особей

промежуточной популяции. Лучшие особи из промежуточной популяции образуют новое поколение, используемое на следующей итерации алгоритма. Псевдокод такой реализации порождения новой популяции приведен ниже.

```
newP=∅; // промежуточная популяция
for( k=0 ; k<ЧИСЛО_НОВЫХ_ОСОБЕЙ ; k++ )
{
   выбрать_две_последовательности_s1_в_P();
   newP=newP∪s1;
   применить_операцию_скрещивания_к_s1();//генерируются две
особи s2
   применить_операцию_мутации_к_s2_с_вероятностью_Pm();
   newP=newP∪s2;
}
вычислить_оценочные_функции_особей_из_newP();
P=( лучшие MAX_ОСОБЕЙ из newP ),
```

где MAX_ОСОБЕЙ – размер популяции, P – текущая и новая популяция, newP – промежуточная популяция, P_m – заданная вероятность мутации.

Рассмотрим недостатки этого подхода. Для того чтобы узнать, какие особи являются лучшими в промежуточной популяции (newP), необходимо вычислить оценочные функции особей этой популяции. При этом вычисление оценочных функций производится для всей промежуточной популяции. Таким образом, вычисление для старых особей будет проведено повторно. Дополнительно, если размер промежуточной популяции (MAX_ОСОБЕЙ + ЧИСЛО_НОВЫХ_ОСОБЕЙ) превышает 32 особи, то вычисление оценочных функций всех особей в новой популяции необходимо организовать два (или более) вызовов программы моделирования, что связано с разрядностью инструментальной ЭВМ.

Чтобы избежать указанных недостатков, предлагается следующий механизм порождения новой популяции, который также основан на параллельном вычислении оценочных функций. После выбора двух особей в качестве родителей они остаются в предыдущей популяции P. В промежуточную популяцию newP после операций скрещивания и мутации будут помещены их потомки, тогда как сами родители – нет. Далее, для получения нового поколения путем выбора лучших особей из старой популяции P и промежуточной newP необходимо будет вычислить оценочные функции только для особей из newP. Это выполняется с помощью программы параллельного моделирования, описанной в предыдущем разделе. При этом также эффективно будут использованы все разряды машинного слова, поскольку число особей в промежуточной популяции newP равно MAX_ОСОБЕЙ и обычно не превышает 32. Псевдокод данной реализации записан ниже.

```
newP=∅;
for( k=0 ; k<MAX_ОСОБЕЙ; k++ ) {
   выбрать_две_последовательности_s1_в_P();
   применить_операцию_скрещивания_к_s1(); // генерируются две особи s2
   применить_операцию_мутации_к_s2_с_вероятностью_Pm();
   newP=newP∪s2;
}
вычислить_оценочные_функции_особей_из_newP(); // только MAX_ОСОБЕЙ
P=( лучшие MAX_ОСОБЕЙ из newP и P).
```

Заметим также, что при таком способе построения новой популяции:

- гарантировано не будут утеряны особи с высокой оценочной функцией, которые, тем не менее, не были выбраны в качестве родителей новых особей;
- вероятность выбора особей с высокой оценочной функцией в качестве родителя повышается в MAX_ОСОБЕЙ раз, поскольку они не удаляются из предыдущей популяции.

Экспериментальные данные

Алгоритм построения тестовых последовательностей на основе генетической стратегии и предложенные методы были реализованы программно на языке программирования C++ в среде визуального программирования C++ Builder. Программа генерации тестовых наборов вошла составной частью в автоматизированную систему моделирования и диагностики АСМИД-Е. Система АСМИД-Е разработана в Институте прикладной математики и механики НАН Украины и выполняет следующие функции: 1) графический или текстовый ввод описания моделируемого устройства; 2) синтаксический анализ описания, необходимый для выявления ошибок подготовки данных; 3) трансляцию описания ЦУ во внутреннюю структуру данных; 4) логическое моделирование исправных ДУ на входных воздействиях, заданных пользователем; 5) генерацию тестов; 6) логическое моделирование неисправных ДУ с целью анализа полноты тестовых наборов.

Апробация работы алгоритма проводилась на схемах из международного каталога ISCAS-89 [8]. Данный каталог содержит набор последовательностных схем с различными характеристиками, предложенных в качестве стандарта для апробации новых методов моделирования и построения тестов. При проведении машинных экспериментов использовалась следующая стратегия. Генерация тестовых наборов проводилась в среде АСМИД-Е с помощью программы, реализующей генетический алгоритм построения тестовых наборов. Данные о наблюдаемости линий схемы вычислялись на

этапе трансляции описания схемы статистическим методом [9] и считались известными программе генерации тестов. Программа выполнялась со следующими значениями параметров: число особей в популяции MAX_ОСОБЕЙ=16, число поколений MAX_ПОКОЛЕНИЙ=16, вес позиции вектора в тесте LH=0.98, вероятность мутации одиночного бита $P_{мутации}$=0.5%. Результаты работы программы: полнота полученных тестовых наборов, время генерации и длина тестов приведены в таблице.

Таблица

Экспериментальные данные работы генетического алгоритма

Имя схемы	Всего неисправностей/ проверилось неисправностей	Полнота / усл. полнота %	Время генерации мин.:сек.	Длина теста
S298	308 / 254	82.79 / 85.39	4:52	637
S344	342 / 323	96.20 / 97.95	3:24	192
S349	350 / 329	95.71 / 97.43	1:09	295
S386	384 / 263	68.49 / 68.49	3:48	606
S641	467 / 404	86.30 / 87.79	3:31	727
S713	581 / 469	80.38 / 81.76	2:56	677
S1196	1242 / 1197	96.38 / 96.38	3:51	1911
S1238	1355 / 1206	90.18 / 90.18	5:52	1183
S1488	1486 / 1047	70.46 / 70.59	14:23	1975
S1494	1506 / 1063	70.58 / 71.18	14:19	1712
S3271	3270 / 3204	97.98 / 97.98	12:45	1125

Полученные экспериментальные данные показывают высокие эксплуатационные характеристики программной реализации предложенного метода генерации тестов.

Литература

1. Niermann T., Patel J.H. HITEC: A Test Generation Package for Sequential Circuits // Proc. European Design Automation Conf. − 1991. − P. 214-218.
2. Sellers F.F., Hsiao M.Y., Bearnson L.W. Analysing errors with the boolean difference // IEEE Transactions on Computers. − 1967. − №5. − P. 675-680.
3. Lisanke R., Brgles F., Degens A..J., Gregory D. Testability-driven random test pattern generation // IEEE Trans. Computer-Aided Design. − 1987. − №6. − P. 1082-1087.
4. Cheng W.T., Patel J.H. PROOFS: A Super Fast Fault Simulator for Sequential Circuits // Proc. The European Conference on Design Automation.– 1990. – P. 475-479.
5. Goldberg D.E., Genetic Algorithm in Search, Optimization, and Machine Learning. – Addison-Wesley. − 1989.
6. 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.
7. Иванов Д.Е., Скобцов Ю.А., Параллельное моделирование неисправностей для последовательностных схем // Искусственный интеллект. − 1999. − №1. − С. 44-50.
8. Brgles F., Bryan D., Kozminski K. Combinational profiles of sequential benchmark circuits // International symposium of circuits and systems, ISCAS-89.– 1989.– P. 1929-1934.
9. Барашко А.С., Скобцов Ю.А., Сперанский Д.В. Моделирование и тестирование дискретных устройств. − К.: Наук. думка, 1992. − 288 с.

Материал поступил в редакцию 14.10.99.