УДК 681.518

Иванов Д.Е.

ГЕНЕТИЧЕСКИЙ АЛГОРИТМ ПОСТРОЕНИЯ ДИАГНОСТИЧЕСКИХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ ЦИФРОВЫХ УСТРОЙСТВ

В статье предлагается новый подход к построению диагностических (локализующих неисправность) последовательностей. Данный подход является двухуровневым. На верхнем уровне из всех текущих множеств неразличимых неисправностей происходит выбор одного целевого множества. На нижнем уровне с помощью генетического алгоритма для выбранного множества строится диагностическая последовательность. Благодаря такой структуре алгоритм позволяет разработчику находить компромисс между затрачиваемым временем и точностью работы

1. Введение.

Наиболее сложными и трудоёмкими при разработке систем автоматизированного диагностирования являются задачи построения тестов. Чаще всего при построении тестов говорят о проверяющих тестах, которые позволяют проверить исправность испытываемого дискретного устройства (ДУ). Задача в этом случае сводится к проверке отличимости некоторого исправного ДУ от неисправного.

Гораздо реже исследуется задача построения диагностических тестов, которые позволяют дополнительно локализовать неисправность. Это связано с тем, что задача являет существенно более сложной. Фактически она заключается в разбиении всего класса исследуемых неисправностей на группы неразличимых неисправностей [1].

Наиболее часто при решении задачи диагностирования используется метод словарей неисправностей. Он заключается в подаче тестовой последовательности на входы схемы, наблюдении её выходов и сравнении полученных значений со значениями, которые сохранены в словаре [1]. Ряд работ посвящён построению таких входных последовательностей [2], поскольку именно они существенно влияют на эффективность данного подхода. Описанный в [2] подход является детерминированным, т.е. гарантирует построение диагностической последовательности, если она существует. Однако недостатком таких методов является их ограниченная применимость к схемам большой размерности, что связано либо с невозможностью построения полного дерева решений, либо с невозможностью его обхода за приемлемое время.

Преодолеть ограничения структурных детерминированных методов помогли генетические алгоритмы (ГА) [3-4]. Фактически, при их построении происходит замена задачи синтеза задачей анализа: для оценки качества текущих последовательностей и продуцирования новых используются результаты моделирования. Генетические алгоритмы обладают наилучшими глобальными поисковыми свойствами и позволяют находить субоптимальные решения за приемлемое время. При этом они относятся к классу недетерминированных алгоритмов в том смысле, что не гарантируют нахождение именно оптимального решения.

Авторы также использовали генетические алгоритмы при решении задач технической диагностики. В частности, были предложены эффективные алгоритмы построения тестов [5], инициализирующих последовательностей [6] и последовательностей верификации эквивалентности [7].

В данной статье, основываясь на предыдущем опыте, авторы предлагают новый алгоритм построения диагностических последовательностей цифровых устройств. Его отличительными чертами являются двухуровневая организация и использование ГА на нижнем уровне для построения диагностической последовательности для некоторого класса неразличимых неисправностей.

Данная статья имеет следующую структуру. Во введении показана актуальность и связь с другими работами в этом направлении. Во втором разделе дана поста-

новка задачи и необходимые определения. В третьем разделе описывается двухуровневая структура алгоритма построения диагностических последовательностей. Четвёртый раздел посвящён описанию генетического алгоритма построения диагностической последовательности для выбранного класса неисправностей. В заключении делаются выводы и указываются направления дальнейших исследований.

2. Задача построения диагностических последовательностей.

В качестве модели ДУ мы будем использовать синхронные последовательностные схемы (рис.1). В данной модели выделяют комбинационные блоки и блоки элементов состояний, которые представлены D-триггерами. Комбинационные блоки представляют собой в общем случае сеть, вершинами которой являются логические элементы, внешние входы и выходы.

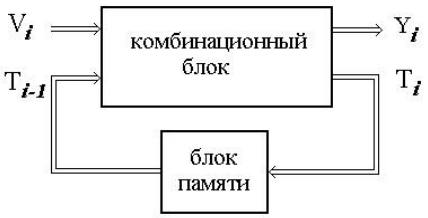

Рис.1. Модель синхронной последовательностной схемы.

В качестве модели неисправности мы используем одиночные константные неисправности. Отметим, что в приведённом виде и без потери общности алгоритм может быть применён к другим распространённым типам неисправностей. От описанного в данной статье новый алгоритм будет отличаться лишь процедурами диагностического моделирования.

Определение 1. Пусть задана исправная цифровая схема A_0 и класс неисправностей $F = \{f_1,...,f_n\}$, который соответственно порождает класс неисправных устройств $A = \{A_1,...,A_n\}$. Входная последовательность T, которая может отличить поведение произвольного устройства A_i от поведения исправного A_0, а также поведения всех остальных неисправных устройств, называется диагностической.

Видно, что в такой постановке задача является более сложной в сравнении с задачей построения проверяющих тестов, в которой достаточно было отличить поведение неисправного устройства A_i от исправного A_0. Фактически, диагностическая последовательность позволяет разбить весь класс неисправностей на классы эквивалентных неисправностей, мощностью каждого из которых равна 1 [8-9]. Такую постановку задачи также часто называют локализацией неисправностей, подчёркивая тот момент, что выделение неисправности в отдельный класс неотличимости точно указывает на место (линию в схеме) возникновения неисправности.

Дадим несколько определений, на основе которых будет строиться решение задачи.

Определение 2. Для заданной схемы A и двух неисправностей f_1 и f_2 последовательность T называется различающей, если выходные реакции $A_1(T)$ и $A_2(T)$ различны хотя бы для одного входного вектора.

Определение 3. Все неисправности $f_1,...,f_n$, которые не различаются заданной входной последовательностью T для заданной схемы A, принадлежат к одному классу неразличимости относительно последовательности T.

Определение 4. Для заданной схемы A и заданного класса неисправностей $F = \{f_1,...,f_n\}$ последовательность T называется различающей, если существует хотя бы одна неисправность $f_j \in F$ такая, что $Aj(T) \neq A_i(T)$ (для $i = \overline{1,n}$ кроме $i = j$) хотя бы для одного входного вектора.

Последнее определение показывает, что при существовании последовательности T с указанными свойствами, класс неисправностей F разбивается на два класса неразличимых неисправностей. Также такую последовательность T часто называют

разбивающей, поскольку она делит один класс неисправностей на несколько. Неформально такую последовательность также можно назвать диагностической. Также последнее определение является конструктивным. Именно на его основе будет строиться наш алгоритм.

3. Общая структура алгоритма построения диагностических тестов.

Цель алгоритма – построение такой тестовой последовательности, которая способна разбить исходный класс одиночных константных неисправностей на максимальное число классов неразличимых неисправностей.

Вначале алгоритма все неисправности принадлежат одному классу неразличимости. Необходимо построить различающую последовательность, которая позволяет разбить этот класс на два или более класса неразличимых неисправностей. Далее необходимо для каждого из полученных классов строить различающие последовательности и т.д. На конечном этапе работы алгоритма последовательности будут строиться для классов, которые содержат только по две неисправности.

На практике результатом работы алгоритма является набор разбивающих последовательностей, с помощью которых полный список неисправностей разбивается на неразличимые классы, каждый из которых может содержать более одной неисправности. Это значит, что алгоритму не удалось построить разбивающую последовательность для неисправностей, которые входят в данный класс. Однако это не говорит о том, что такая последовательность не существует в принципе. В этом смысле предлагаемый алгоритм относится к классу недетерминированных.

Структурно алгоритм представляет собой итеративный процесс генерации диагностических последовательностей. Всякий раз, когда удаётся построить последовательность, которая разбивает выбранный класс неразличимых неисправностей на несколько классов, она добавляется в финальный тест.

Каждая итерация алгоритма состоит из трёх фаз.

1) выбор целевого класса из текущего множества классов неразличимости;
2) построение различающей последовательности для выбранного класса;
3) если в фазе 2 построена различающая последовательность, то выполняется моделирование для других классов, чтобы проверить возможность того, что она также является расщепляющей для них.

При этом фаза 1 является подготовительной для фазы 2, а фаза 3 – заключительной. Таким образом, указанные фазы 1 и 2 представляют собой верхний уровень алгоритма, а фаза 2 – нижний, причём именно последний основан на ГА. Видно, что такая структура алгоритма в общем виде повторяет структуру алгоритма генерации тестов для одиночных константных неисправностей [5]. Это позволяет говорить о том, что они оба относятся к классу двухуровневых недетерминированных алгоритмов. Особенностью данного класса является то, что генетический алгоритм используется только на нижнем уровне, тогда как цель его работы выбирается на верхнем уровне с помощью некоторых эвристик.

Укрупнённый псевдокод алгоритма приведён ниже.

```
Построение_диагностического_теста(схема)
{
  чтение_описания_схемы();
  построение_полного_списка_неисправностей_F();
  while( не_достигнут_критерий_остановки() )
  {
    // фаза 1 - выбор целевого множества
    F_j=выбрать_целевое_множество();//случайная генерация
    if( F_j==NULL )
    {
      return;//целевое множество не выбрано-конец алгоритма
```

```
    }
    иначе
    {
      // фаза 2 – генетический алгоритм
      Ti=ГА_построения_ДП(схема, Популяция, множество Fj);
      if( Ti диагностическая для Fj )
      {
        добавить_в_финальный_тест(Ti);
        // фаза 3 – дополнительная проверка
        дополнительная_проверка_для_всех_других_множеств(Ti);
      }// конец if – вызов Фазы 3 алгоритма
    }  // конец if – выбрана целевое множество
  }    // конец while – не достигнут критерий остановки
}      // конец алгоритма
```

Опишем подробнее общую структуру алгоритма.

Фаза 1 алгоритма представлена процедурой «Выбрать_целевое_множество». В ней происходит генерация набора случайных входных последовательностей $\{T_i\}$. Далее производится моделирование на каждой из данных последовательностей для всех классов неразличимости F_j, которые построены на текущий момент. Для каждого из них происходит вычисление оценочной функции $E_j(T_i, F_j)$ (см. следующий раздел). Если в данном месте ни для одной из последовательностей не получено значение, которое превышает некоторое пороговое: $E_j(T_i, F_j) < E_{пор}$, то происходит генерация нового множества последовательностей $\{T_i\}$. Такая попытка производится не более, чем предопределённое число раз.

Если же для некоторого класса неразличимости F_j получено значение $E_j(T_i, F_j) > E_{пор}$, то данный класс выбирается в качестве целевого для фазы 2 алгоритма, причём последний набор последовательностей $\{T_i\}$ выбирается в качестве начальной популяции генетического алгоритма.

В фазе 2 алгоритма для выбранного класса неразличимости F_j строится, по возможности, диагностическая последовательность. Этап основан на генетическом алгоритме и подробно описан в следующем разделе.

Фаза 3 алгоритма является дополнительной. В ней на построенной диагностической последовательности производится моделирование для всех неразличимых классов неисправностей (кроме F_j - целевого класса фазы 2), которые построены к текущему моменту времени. Цель данного моделирования проверить: является ли построенная для F_j диагностическая последовательность разбивающей для других классов неразличимости.

Фазы 1-3 алгоритма повторяются итеративно, но не более чем заранее заданное число раз.

Видно, что данный алгоритм использует большое число эвристик и для подтверждения его эффективности требует их тщательного подбора.

4. Генетический алгоритм построения диагностических тестов.

В данном разделе будет описан генетический алгоритм построения различающей последовательности для выбранного множества неисправностей. Известно [4], что для построения генетического алгоритма необходимо описать его основные компоненты: кодирование особей и популяций, оценочную функцию для особи, генетические операции (выбор, скрещивание, мутация, построение новой популяции).

Поиск алгоритмом производится в пространстве всех возможных входных последовательностей с заданной шириной входного вектора, которая равна числу внешних входов схемы. Кодирование особи является классическим в задачах построения входных последовательностей [5]. Особь это последовательность, которая состоит из некоторого числа входных двоичных наборов. Набор таких особей представляет собой популяцию.

Оценочная функция является одним из важнейших элементов всего алгоритма. Эффективная оценочная функция позволяет находить необходимые решения. С другой стороны скорость её вычисления влияет на скорость всего алгоритма.

Очевидно, что для некоторой входной последовательности T_j и класса неисправностей F_i достаточно легко определить является ли она диагностической, т.е. разбивает ли она данный класс как минимум на два класса неразличимости. Для этого достаточно выполнить моделирование поведения схем для всех неисправностей, которые входят в F_i. Однако оценить, насколько близко данная последовательность T_j подошла к решению такой задачи, гораздо труднее. В задачах построения входных идентифицирующих последовательностей [10] оценочная функция часто строится на основании показателя активности схемы: чем больше различных значений в двух сравниваемых схемах, тем выше оценка данной входной последовательности. В описываемом алгоритме оценочная функция вычисляется на основе двух следующих параметров:

- число триггеров с различными значениями в неисправных схемах, которые соответствуют неисправностям, входящим в текущий класс неразличимости;
- число выходов вентилей с различными значениями в неисправных схемах, которые соответствуют неисправностям, входящим в текущий класс неразличимости.

Для вычисления оценочной функции всей последовательности сначала вычисляются оценочные функции каждого входного вектора, который входит в данную последовательность. Для класса неразличимости F_i и k-го вектора последовательности T_j данная функция имеет следующий вид

$$E(T_j^k, F_i) = k_1 \cdot \sum_{n=1}^{N_{тр}} R_n^1(T_j^k, F_i) + k_2 \cdot \sum_{n=1}^{N_{вент}} R_n^2(T_j^k, F_i), \qquad (1)$$

где:

- $R_n^1(T_j^k, F_i)$ - функция различия выходов триггеров;
- $R_n^2(T_j^k, F_i)$ - функция различия выходов вентилей;
- k_1 и k_2 нормирующие коэффициенты, показывающие, что различия значений на выходах триггеров важнее различия на выходах элементов;
- $N_{тр}$ и $N_{вент}$ - число триггеров и выходов элементов в схеме соответственно.

Функции различия определяются следующим образом:

$$R_n^{\alpha}(T_j^k, F_i) = \begin{cases} 0, \textit{если выходы элементов с идексом } n, \textit{которые принадлежат} \\ \quad \textit{классу } F_i, \textit{одинаковы после подачи набора } T_j^k; \\ 1, \textit{иначе}; \end{cases} \qquad (2)$$

где $\alpha = 0{,}1$ для множества триггеров и выходов элементов схемы соответственно.

На основании оценки каждого вектора, входящего в последовательность, вычисляется её оценочная функция, которая имеет вид:

$$E(T_j, F_i) = \max_k (E(T_j^k, F_i)), \quad (3)$$

В качестве оператора селекции используется пропорциональный отбор (метод рулетки) без ранжирования.

В качестве скрещивания используются две операции: вертикальное (по входам схемы) и горизонтальное (по тактам времени) скрещивание.

Операторы мутации также являются стандартными для данного кодирования и представлены следующими операциями:

- добавление вектора в случайную позицию;
- удаление вектора из случайной позиции;
- изменение случайного вектора.

Для построения промежуточной популяции итеративно выбираются пары родителей, над которыми с заданными вероятностями $P_{скр}$ (для операции скрещивания) и $P_{мут}$ (для мутации) выполняются соответствующие операции. Результирующие особи-потомки помещаются в промежуточную популяцию.

При построении новой популяции из текущей и промежуточной используется стратегия элитизма, причём в популяции обновляется 80% особей.

В виде псевдокода описанный алгоритм приведён ниже.

```
ГА_построения_ДП(схема, Популяция, множество F)
{
  for( i=0 ; i<MAX_ПОКОЛЕНИЙ ; i++ )
  {
    for( каждой особи T в Популяции)
      вычислить E(T,F);
    for( k=0 ; k<ЧИСЛО_НОВЫХ_ОСОБЕЙ ; k++ )
    {
      выбрать_две_особи_в_популяции();
      выполнить_скрещивание();
      выполнить_мутацию ();
      добавить_в_промежуточную_популяцию();
    };
    построить_новую_популяцию();
    for( каждой особи T в Популяции )
    {
      if( T диагностическая для F )
        return T;
    };
  };
  return( НЕТ_ПОСЛЕДОВАТЕЛЬНОСТИ );
}
```

Вычисление описанной выше оценочной функции в генетических алгоритмах традиционно основано на моделировании. Обычно для этих целей используется адаптированная версия алгоритмов моделирования типа PROOFS [11] либо ему подобных [12]. Однако в данном методе прямое использование таких алгоритмов невозможно. Это связано с тем, что указанные алгоритмы моделирования ориентированы на быструю проверку полноты заданной входной последовательности. В них происходит сравнение поведения всех неисправных схем с исправной схемой. Более того, если для неисправности в некоторый момент времени установлено, что она уже проверена дан-

ной последовательностью, то она вычёркивается из списка и моделирование поведения неисправной схемы для неё далее не производится. Адаптация алгоритма моделирования для вычисления оценочной функции диагностических тестов заключается в следующих изменениях:

- неисправность удаляется из списка неисправностей только, когда она будет отличена от всех остальных, т.е. образует класс неразличимости с размерностью 1;
- поведение схемы вычисляется для всех входных векторов для всех неисправностей из текущего списка;
- после моделирования текущего входного вектора необходимо сравнить значение на внешних выходах схемы со значениями выходов для всех неисправностей, которые принадлежат одному классу неразличимости.

Также заметим, что при адаптации алгоритма используются дополнительные структуры данных, которые фиксируют распределение неисправностей по текущим классам неразличимости.

5. Выводы.

В статье впервые предложен двухуровневый недетерминированный алгоритм построения диагностических последовательностей цифровых устройств. В основе данного алгоритма лежит генетический подход и алгоритм диагностического моделирования. Отсутствие необходимости явного построения всего дерева решений позволяет данному алгоритму работать с последовательностными схемами большого размера. Предложенный алгоритм относится к классу двухуровневых алгоритмов, в котором на верхнем уровне происходит выбор цели, а на втором уровне уже с помощью генетического алгоритма строится необходимая последовательность. Такая структура позволяет разработчику в широких пределах регулировать время работы алгоритма и его точность.

В качестве дальнейших исследований можно отметить следующие направления:

- исследование зависимости эффективности алгоритма от значений эвристических констант;
- ввиду очень большого числа процедур моделирования – разработка параллельных многопоточных версий алгоритма.

Литература

1. M. Abramovici, M.A. Brauer, A.D. Friedmann, Digital system testing and testable design, Computer science press, New York, NY (USA), 1990.
2. G. Cabodi, P. Camurati, F. Corno, P. Prinetto, M. Sonza Reorda Sequential circuit diagnosis based on formal verification techniques // Proc. of International Test Conf., 1992.- pp.187-196.
3. Goldberg D.E., Genetic Algorithm in Search, Optimization, and Machine Learning.- Addison-Wesley, 1989.- 432p.
4. Ю.О. Скобцов. Основы эволюционных вычислений.– Донецк: ДонНТУ, 2008.- 326с.
5. Иванов Д.Е., Скобцов Ю.А. Генерация тестов цифровых устройств с использованием генетических алгоритмов // Труды института прикладной математики и механики НАН Украины. – Т.4. – Донецк, ИПММ.– 1999.– С.82-88.
6. Д.Е. Иванов, Ю.А. Скобцов, А.И. Эль-Хатиб Построение инициализирующих последовательностей синхронных цифровых схем с помощью генетических алгоритмов.- Проблеми інформаційних технологій.- 2007.- №1.- С.158-164.
7. Иванов Д.Е. Генетический алгоритм проверки эквивалентности последовательностных схем // «Радіоелектроніка. Інформатика. Управління».- Запоріжжя, ЗНТУ.- 2009.- №1(20).- С.118-123.
8. P. Camurati, D. Medina, P. Prinetto, M. Sonza Reorda A diagnostic test pattern generation algorithm // Proc. of International Test Conf.- 1992.- pp.52-58

9. T. Grüning, U. Mahslstedt, H. Koopmeiners DIATEST: a fast diagnostic test pattern generator for combinational circuits // Proc. Int. Conf. on Computer Aided Design.- 1991.- pp.194-197
10. Иванов Д.Е. Генетические алгоритмы построения идентифицирующих последовательностей для цифровых схем с памятью // Наукові праці Донецького національного технічного університету. Серія: "Обчислювальна техніка та автоматизація". Випуск 14(129).- Донецьк: ДонНТУ.– 2008.- С.97-106.
11. Niermann T.M., Cheng W.-T., Patel J.H. PROOFS: A Fast, Memory-Efficient Sequential Circuits Fault Simulator // IEEE Trans. CAD.– 1992.– P.198-207.
12. Иванов Д.Е., Скобцов Ю.А. Параллельное моделирование неисправностей для последовательностных схем // Искусственный интеллект.– 1999.- №1.– С.44-50.