УДК 681.518

Д.Е. Иванов, Р. Зууи

АЛГОРИТМ СИМУЛЯЦИИ ОТЖИГА ПОСТРОЕНИЯ ТЕСТОВ ЦИФРОВЫХ УСТРОЙСТВ

Введение и постановка задачи. Тестирование цифровых схем на различных этапах жизненного цикла их проектирования остаётся одной из центральных задач в технической диагностике. Это связано с непрекращающимся ростом размерности современных схем. Тестирование состоит в подаче входных воздействий на внешние входы схемы и наблюдении внешних выходов. Различие в поведении эталонной схемы с текущей проверяемой, которое наблюдается как несовпадение сигналов на внешних выходах, показывает неисправность схемы. Трудность задачи связана с тем, что число схем, которые должны сравниваться с эталонной очень велико. Например, для обычно используемой модели одиночных константных неисправностей число таких схем будет $2*N$, где N - число линий в схеме. Существующие структурные алгоритмы [1] не дают приемлемых результатов для схем с числом вентилей более десяти тысяч и числом триггеров более тысячи.

Для поиска решений в сложных задачах поиска решений в последнее десятилетие стали применять недетеменированные алгоритмы, которые ищут субоптимальные решения. Одним из примеров такого подхода являются генетические алгоритмы [2]. Парадигма впервые предложена в [3]. Наиболее широко её применение в автоматизированном проектировании цифровых схем разрабатывается в политехническом университете г.Турина, Италия [4]. Генетические алгоритмы относятся к направленным случайным методам поиска. Преимущество их применения состоит как в прозрачности стратегии, так и в возможности работать со схемами большой размерности, для которых структурные методы не дают результата. Эта возможность появляется в связи с тем, что необходимая для построения тестов информация получается на основе моделирования поведения схем. Здесь применяется как моделирование работы исправных схем, так и схем с неисправностями. К недостаткам такого подхода можно отнести достаточно продолжительное время работы, что связанно именно с оценкой качества входных последовательностей, которая выполняется на основе моделирования.

Среди других оптимизационных стратегий можно выделить алгоритм симуляции отжига (СО) [5]. Данная стратегия также относится к эволюционным, однако использует иную парадигму. Решение ищется путём улучшения свойств одного первоначально сгенерированного решения. Авторы уже применяли данную стратегию к задаче построения инициализирующих последовательностей [6]. В этой работе было показано, что эффективность алгоритма на основе стратегии симуляции отжига не уступает эффективности генетического алгоритма. Вместе с тем, построение алгоритма и его программная реализация гораздо проще.

В данной статье авторы предлагают новый двухуровневый алгоритм построения тестов для последовательностных схем, в котором для построения теста одной неисправности используется стратегия СО.

Данная статья имеет следующую структуру. Во втором разделе будет кратко описана оптимизационная стратегия симуляции отжига. В третьем разделе мы опишем общую двухуровневую структуру алгоритма генерации тестов и его верхний уровень. В следующем разделе будет представлен алгоритм симуляции отжига построения теста для заданной целевой неисправности. В пятом разделе будут приведены результаты машинных экспериментов и их сравнение с другими подходами. В заключении сформулированы результаты и направления, которые можно выбрать в качестве дальнейших исследований.

Стратегия симуляции отжига. Стратегия симуляции отжига впервые была предложена Метрополисом в [7] для нахождения состояний группы молекул в остывающем слитке. Из данной статьи заимствованы все основные термины стратегии: начальная и конечная температура, скорость остывания, конфигурация и т.д. Только в 1983 году Кирпатриком в [5] было показано, что описанный подход может быть с успехом применён к задачам оптимизации в широком смысле. В частности, автор описывает примеры решения следующих задач автоматизированного проектирования: разбиения схемы на подсхемы, размещения чипов на плате, трассировка соединительных линий.

Таким образом, стратегия симуляции отжига является ещё более «молодой», чем генетические алгоритмы. Исследования в этой области ещё только начинаются и разработано относительно небольшое число приложений к практическим задачам. В основном это классические задачи комбинаторной оптимизации, которые носят иллюстративный характер, показывая саму возможность применения стратегии на практике. К ним относятся как упомянутые выше задачи, так и задача построения расписаний [8], задача коммивояжёра [9] и т.д..

Кратко опишем общий алгоритм стратегии симуляции отжига. Как и в генетических алгоритмах, применяется кодирование потенциальных решений в пространстве поиска. Закодированное решение

называется конфигурацией K. Для оценки качества решения задачи в точке K задаётся функция оценки $C(K)$. Математически задача формулируется как поиск оптимума функции оценки:

$$C(K_i) \to \max . \quad (1)$$

Сам алгоритм поиска оптимальной конфигурации представляет собой итеративный процесс построения новой конфигурации из текущей. Итерации прекращаются либо когда найдено решение, либо когда превышено заданное число итераций.

Один шаг итерации выполняется для текущей температуры T_i и заключается в следующем. Для текущей конфигурации K_i с помощью заданных операций возмущения строится окружение, которое состоит из некоторого множества конфигураций:

$O(K_i) = \{K_1^{'}, K_2^{'}, ..., K_m^{'}\}$.

Размер m окружения $O(K_i)$ задаётся как параметр алгоритма. Каждая точка окружения $K_i^{'}$ является небольшой модификацией текущей конфигурации K_i. Смысловая нагрузка оператора возмущения задаётся исходя из самой задачи. Далее необходимо вычислить оценку для каждой конфигурации из окружения $O(K_i^{'})$, $i = \overline{1,m}$. Если оценка конфигурации улучшилась $C(K_i^{'}) > C(K_i)$, то происходит замена текущей конфигурации на лучшую: $K_i = K_i^{'}$. Если же оценка ухудшилась, то принимать или не принимать такие ухудшения определяется распределением Больцмана. При этом вероятность замены текущей конфигурации на более худшую определяется выражением:

$$P = \exp(-(C(K_i^{'}) - C(K_i))/kT_i), \quad (2)$$

где k - константа Больцмана, T_i - текущая температура итерации. Алгоритм строится таким образом, чтобы с увеличением номера итерации вероятность P приёма ухудшенных конфигураций уменьшалась со временем. Для этого задаётся распределение температур по итерациям $\{T_i\}$, которое обычно заключается в постепенном уменьшении температуры. Таким образом, при большей температуре вероятность P будет больше, чем при низкой. Этот момент позволяет алгоритму СО избегать локальных максимумов и является аналогом мутации в генетических алгоритмах. Скорость уменьшения температуры также ключевой момент алгоритма и существенно влияет на поиск оптимума.

Общая схема алгоритма построения тестов. В качестве модели в алгоритме используется модель синхронных последовательностных схем [10]. В данной модели происходит преобразование схемы в комбинационный эквивалент путём удаления линий обратной связи. При этом входы элементов состояний, которые представлены D-триггерами, становятся псевдовыходами схемы, а их выходы – псевдовходами. Моделирование поведения такой схемы происходит итеративно по тактам модельного времени. Причём значения сигналов, которые сформировались на линиях внешних псевдовыходов, попадают на псевдовходы одновременно с подачей следующего входного набора (по тактам синхронизации).

Задача построения тестов для синхронных последовательностных схем в сравнении с комбинационными является существенно более сложной. Это определяется следующими факторами:

- заранее не известна длина требуемой последовательности;
- влияние неисправности на поведение схемы сначала необходимо распространить на внешние псевдовыходы схемы;
- после этого влияние неисправности следует распространить на внешние выходы.

Алгоритм начинает работу с подготовительных действий: чтения описания схемы из файла и построения полного сжатого по эквивалентности списка неисправностей [11]. После этого открывается главный цикл алгоритма. Он выполняется, пока не выполнены условия остановки, в качестве которых выступают следующие:

- в списке больше нет непроверенных неисправностей;
- время работы алгоритма превысило предварительно заданное значение;
- достигнуто заданное предельное число итераций алгоритма.

В начале очередной итерации алгоритм выбирает некоторую ещё не проверенную тестом неисправность – фаза 1 алгоритма. Данная неисправность будет служить в качестве целевой неисправности в фазе 2 алгоритма, в которой происходит вызов непосредственно процедуры симуляции отжига построения теста. Если неисправность обнаружена в фазе 2 некоторой последовательностью, то для данной последовательности выполняется дополнительное моделирование (фаза 3), цель которого заключается в проверке: обнаруживает ли данная последовательность ещё некоторые неисправности

кроме целевой. После этого последовательность добавляется в тест. Если же в фазе 2 не удалось построить тестовую последовательность для неисправности $f_{цел}$, то данная неисправность отмечается как непроверяемая, чтобы исключить её выбор в качестве целевой на следующих итерациях алгоритма. Укрупнённый псевдокод описанного алгоритма приведён ниже.

```
CO_построения_тестов(схема)
{
  чтение_описания_схемы();
  построение_полного_списка_неисправностей_F ();
  while( не_достигнут_критерий_остановки() )
  {
    // псевдослучайная генерация – Фаза 1 алгоритма
    f_цел =выбрать_целевую_неисправность_из_списка(F);
    if( f_цел ==NULL ) { возврат };
    иначе
    {
      // применение алгоритма CO – Фаза 2 алгоритма
      s_тест =вызов_CO_построения_теста(схема, f_цел , s_акт );
      if( f_цел проверена последовательностью s_тест )
      {
        // дополнительное моделирование – Фаза 3
        моделирование_с_неисправностями(схема, s_тест , F );
        добавить_в_тест( s_тест );
      }// конец if – вызов Фазы 3 алгоритма
      else
      {
        отметить_неисправность_как_непроверяемую( f_цел );
      }// конец else – тест не построен
    }  // конец if – выбрана целевая неисправность
  }    // конец while – не достигнут критерий остановки
}      // конец алгоритма
```

Из псевдокода видно, что непосредственно процесс построения тестовых последовательностей разбит на два взаимосвязанных уровня. На верхнем уровне генерация тестов для всего списка неисправностей происходит с помощью псевдослучайного метода. На нижнем уровне для построения теста для одной неисправности используется алгоритм CO.

Остановимся подробнее на описании фаз 1 и 3, представляющие верхний уровень алгоритма, а в следующем разделе мы опишем второй уровень (фаза 2), который является центральным.

В фазе 1 алгоритма выбирается некоторая неисправность-цель $f_{цел}$. Для этого происходит итеративная случайная генерация входных последовательностей (конфигураций) $Conf_{тек}$ с длиной L. Для каждой последовательности выполняется моделирование с неисправностями на полном списке ещё непроверенных неисправностей F. Если для некоторой последовательности $Conf_{тек}$ в результате такого моделирования установлено, что она обнаруживает новые неисправности, то она сразу добавляется в финальный тест. Если новых неисправностей не обнаружено, то путём просмотра списка ищется такая неисправность, у которой различия наблюдаются на элементах состояний (псевдовыходах). Если такая неисправность найдена, то она возвращается в качестве целевой $f_{цел}$ в вызывающую функцию и становится основной для второго уровня алгоритма. Если же после просмотра всего списка неисправностей не обнаружено активных неисправностей, то длина последовательности L увеличивается и происходит переход к следующей итерации. Процедура возвратит признак невозможности найти целевую неисправность в том случае, когда счётчик итераций ЧислоПопыток достигнет максимального значения MAX_ЧислоПопыток, которое задаётся предварительно. Псевдокод данного фрагмента алгоритма приведён ниже.

```
Выбрать_целевую_неисправность_из_списка( F )
{
  While( ЧислоПопыток < MAX_ЧислоПопыток )
```

```
  {
    L =ОбновитьДлину();
    Conf_тек =ПостроитьНачальнуюКонфигурацию( L );
    моделирование_с_неисправностями(схема, Conf_тек , F );
    if( обнаружены новые неисправности )
    {
      добавить_в_тест( Conf_тек );
    }
    else
    {
      if( в F есть активизированная неисправность f_i )
      {
        return( f_цел = f_i );
      }  // конец if – есть активизированные неисправности
    }    // конец else – обнаружены новые неисправности
  }      // конец while – по числу попыток
  return NULL; //целевая неисправность не выбрана
}        // конец процедуры
```

Заметим, что такое построение процедуры поиска целевой неисправности является очень эффективным в связке со вторым уровнем. Большая часть всех неисправностей обнаруживается именно в фазе 1.

Алгоритм симуляции отжига построения теста одной неисправности. Фаза 2 является основной в описываемом алгоритме. Именно здесь используется алгоритм симуляции отжига. Чтобы применить общую схему алгоритма СО для решения конкретной задачи необходимо дополнительно задать функцию оценки, операцию возмущения и расписание температур.

В алгоритме используется кодирование конфигураций в виде двоичных последовательностей [4,10], которое является традиционным при решении такого класса задач.

Как было сказано выше, оценка качества конфигурации основана на моделировании с неисправностями. Для некоторой входной последовательности s и фиксированной неисправности $f_{цел}$ она имеет вид:

$$Cost(s, f_{цел}) = \sum_{i=1}^{i=длина} H^i * h(v_i, f_{цел}), \qquad (3)$$

где H – предварительно заданная константа в диапазоне $0 < LH <= 1$, благодаря которой предпочтение отдаётся более коротким последовательностям. Функция $h(v_i, f_{цел})$ определяет качество i-го вектора в последовательности и использует информацию о различии поведения исправной и неисправной функций. Из-за краткости изложения мы не будем останавливаться на данном моменте. Подробно вычисление функций такого вида описано в [10].

Оператор возмущения представлен следующими операциями: удаление вектора, добавление вектора, изменение вектора, изменение столбца, выбор между которыми производится случайно.

Применяется линейное изменение температуры от $T_{нач}$ до $T_{кон}$ с шагом 1.

После задания компонент можно описывать основную структуру алгоритма СО фазы 2. Входом для данного уровня является целевая неисправность $f_{цел}$ (для которой необходимо построить проверяющую последовательность $s_{тест}$), а также активизирующая последовательность $s_{акт}$. Получив исходные данные, процедура в соответствии с алгоритмом СО итеративно для заданного ряда температур производит построение окружения с оценкой всех конфигураций, которые в него попадают. Псевдокод процедуры построения теста для одной неисправности представлен ниже.

```
СО_генерация_теста(схема, f_цел , s_акт )
{
  Conf_тек  = s_акт ;
  T_тек = T_нач ;
  Cost_тек =ОценитьКонфигурацию( f_цел , Conf_тек );
```

```
  while( T_тек > T_кон )
  {
    for( int i=0 ; i<ЧислоИтераций ; i++ )
    {
      Conf_пром =Возмущение( Conf_тек );
      Cost_пром =ОценитьКонфигурацию(схема, f_цел , Conf_пром );
      if( f_цел ->проверяемость == проверна )
      {
        return s_тест = Conf_пром ;
      }
      ΔCost = Cost_пром − Cost_тек ;
      if( (ΔCost < 0) || (rand()>exp(−ΔCost /(k · T_тек))) )
      {
        Conf_тек = Conf_пром ;
      }// конец if – отрицательное изменение стоимости
    }
    T_тек = Обновить_T( T_тек );
  } // конец while – цикла по температуре
  return нет_теста;
}   // конец процедуры
```

Дадим краткие пояснения к псевдокоду. Важнейшей является процедура «ОценитьКонфигурацию()», которая принимает в качестве параметров неисправность $f_{цел}$ и конфигурацию, представленную некоторой входной последовательностью. На основании моделирования с неисправностями на полученной последовательности для исследуемой схемы вычисляется числовая оценка конфигурации в соответствии с формулой (3).

Процедура «Возмущение()» для заданной конфигурации на основании выше описанных операций строит новую конфигурацию, для которой далее производится оценка. Если оценка конфигурации улучшилась либо позволяет распределение Больцмана, то текущая конфигурация заменяется новой. Данный фрагмент выполняется для каждой температуры такое число раз, которое задаётся константой «ЧислоИтераций». Если в какой либо момент новая конфигурация обнаружит целевую неисправность, что видно по результатам моделирования в функции «ОценитьКонфигурацию()», то итерации алгоритма симуляции отжига прекращаются, а данная конфигурация передаётся в качестве результата работы. Расписание температур реализуется процедурой «Обновить_T()».

Экспериментальные данные. В данном разделе мы опишем реализацию, проведённые машинные эксперименты и приведём численные результаты апробации на схемах из каталога ISCAS-89 [12].

Предложенный выше двухуровневый алгоритм СО построения тестов реализован программно на языке C++. В качестве процедур вычисления оценок конфигураций использовались модифицированные процедуры моделирования схем с неисправностями [13]. Общий размер реализации составил примерно 4600 строк кода, включая процедуры моделирования. Эксперименты проводились на инструментальной ЭВМ с процессором Core2Quad Q6600 2.4ГГц, оперативная память 2Гбайта.

Из-за краткости изложения мы не будем приводить описание экспериментов по выбору эвристических констант, а приведём полученные значения: $T_{нач} = 100$, $T_{кон} = 1$, $\Delta T = 1$, константа Больцмана $k = 1e-6$, начальная длина теста =1, число итераций для каждой температуры =50.

Результаты машинных экспериментов приведены в табл.1. Дополнительно таблица содержит результаты предыдущих экспериментов для аналогичного генетического алгоритма [14]. Заметим, что сравнение экспериментальных данных производилось для «настроенных» алгоритмов, когда параметры фиксированы для всего набора схем и не выставляются отдельно для каждой из них. Очевидно, что любая подстройка параметров под конкретную схему может ещё улучшить полноту получаемых тестов, возможно, за счёт глубины поиска и времени работы. Также мы не проводили сравнение временных характеристик алгоритмов, поскольку в работах использовались различные инструментальные платформы.

Результаты сравнения позволяют говорить, что алгоритм СО генерации тестов позволяет получать тестовые последовательности с большей полнотой в сравнении с аналогичным ГА. Однако длина получаемых последовательностей выше во всех проведённых экспериментах, причём в некоторых -

Табл.1. Экспериментальные данные.

Имя схемы	Всего неисправностей	Алгоритм СО			Генетический алгоритм	
		полнота теста	длина теста	время генерации, мин:сек	полнота теста	длина теста
s298	308	83.11	1552	0:33	82.79	637
s344	342	96.20	438	0:16	96.20	192
s349	350	95.71	566	0:07	95.71	295
s386	384	70.57	1189	0:23	68.49	606
s641	467	86.50	1517	0:34	86.30	727
s713	581	81:07	1716	0:42	80.38	677
s1196	1242	97.95	1994	1:43	96.38	1911
s1238	1355	91.27	3866	1:51	90.18	1183
s1488	1486	73.62	2202	7:11	70.46	1975
s1494	1506	73.30	1947	7:45	70.58	1712
s3271	3270	98.13	2807	1:58	97.98	1125
s3330	2870	67.39	8312	1:37:51		
s3384	3380	90.71	3191	37:28		
s5378	4603	67.37	6913	22:00		
s6669	6684	98.34	7640	44:56		
s35932	39094	74.28	597	1:09:54		

существенно. Это можно объяснить тем, что при коллективном поиске решений в ГА из двух особей в популяции с одинаковыми оценочными функциями, всегда будет выбрана та, у которой длина меньше. Сама природа алгоритма СО лишает его такой возможности отбора, поскольку происходит эволюция одного потенциального решения. При этом оказывается, что добавление операции удаления вектора в оператор возмущения не решает данной проблемы, поскольку данная операция всё равно должна балансировать по количеству применений с другими операциями, улучшающими свойства конфигурации. В генетических же алгоритмах все операции мутации имеют большее применение в силу размерности популяции.

Выводы. В статье предложен двухуровневый алгоритм построения тестов для цифровых схем, который использует новую оптимизационную стратегию симуляции отжига. Несмотря на то, что эволюция производится для одного потенциального решения, тщательный подбор параметров процесса симуляции отжига позволил добиться лучших экспериментальных результатов в сравнении с простым генетическим алгоритмом.

В качестве дальнейших исследований можно отметить два основных направления. Поскольку алгоритм на основе стратегии симуляции отжига показал хорошие практические результаты, то есть смысл применить данную стратегию к решению других задач идентификации цифровых устройств. Параллельные версии генетических алгоритмов (реализованные как на многопроцессорных, так и однопроцессорных инструментальных средствах) показывают лучшие результаты в сравнении с традиционными за счёт лучшей структуризации поиска. Исходя из этого, можно строить параллельные версии алгоритмов симуляции отжига с целью улучшить его поисковые свойства.

ЛИТЕРАТУРА

1. S.-Y. Huang, K.-T. Chen, “Formal Equivalence Checking and Design Debugging”, Kluwer Academic Publishers.- Boston.- 1998.- 229p.
2. Ю.А. Скобцов. Основы эволюционных вычислений.– Донецк: ДонНТУ, 2008.- 326с.
3. Goldberg D.E., “Genetic Algorithm in Search, Optimization, and Machine Learning”, Addison-Wesley.- 1989.- 432p.
4. F. Corno, P. Prinetto, M. Rebaudengo, M. Sonza Reorda GATTO: a Genetic Algorithm for Automatic Test Pattern Generation for Large Synchronous Sequential Circuits // IEEE Transactions on Computer-Aided Design, August 1996.- Vol.15, №8.- pp. 943-951.
5. S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi Optimization by simulating annealing // Science, 1983.- №220, pp.671-680.

6. Иванов Д.Е., Зууи Р. Алгоритм построения инициализирующих последовательностей цифровых схем, основанный на стратегии симуляции отжига // Искусственный интеллект, 2009.- №4.- с.415-424.
7. Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. and Teller, E. Equation of state calculations by fast computing machines // J. Chemical Phys., 1953.- Vol.21.- pp. 1087-1092.
8. Bouleimen K., Lecocq H. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple modes version // Eur. J. Oper. Res., 2003.- №149:2.- pp.268–281.
9. Bonomi, E. and Lutton, J. L. The N-city traveling salesman problem: Statistical mechanics methods and Metropolis algorithm // SIAM Rev., 1984.- Vol.36.- pp. 551-568.
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. Барашко А.С., Скобцов Ю.А., Сперанский Д.В. Моделирование и тестирование дискретных устройств. – Киев: Наукова думка, 1992. – 288 с.
12. Brgles F., Bryan D., Kozminski K. Combinational profiles of sequential benchmark circuits // International symposium of circuits and systems, ISCAS-89.– 1989.– p.1929-1934.
13. С.А.Закусило, Д.Е.Иванов, В.Ю.Скобцов, Ю.А.Скобцов. Генетический подход к генерации проверяющих тестов в системе АСМИД-Е // Труды конференций "Искусственные интеллектуальные системы" и "Интеллектуальные САПР".- Москва:Физматлит.- 2002.- С.49-55.
14. 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.

АЛГОРИТМ СИМУЛЯЦІЇ ВІДЖИГУ ПОБУДОВИ ТЕСТІВ ЦИФРОВИХ ПРИСТРОЇВ

Д.Є. Іванов, Р. Зуаyі

В даній статі пропонується новий дворівневий алгоритм рішення класичної задачі генерації тестів для синхронних послідовністних схем. Він заснований на новій оптимізуючій стратегії симуляції віджигу. Побудова тесту для деякого обраного пошкодження відбувається шляхом покращення властивостей одного попереднього рішення. Дані для оцінювання тестових властивостей цього рішення отримуються на основі моделювання з пошкодженнями схеми, що досліджується. Такий підхід дозволяє ефективно будувати тести для схем середньої та великої розмірності, що підтверджується результатами апробації на схемах з міжнародного каталогу ISCAS-89.

Ключові слова: цифрова послідовністна схема, генерація тестів, симуляція віджигу, моделювання з пошкодженнями.

THE SIMULATING ANNEALING ALGORITHM FOR THE TEST SEQUENCE GENERATION

D.E. Ivanov, R. Zouaoui

In this paper a new two-level algorithm for solving the classical problem of test generation for the sequential circuits is proposed. This algorithm is based on the new optimizing strategy named simulating annealing. Test construction for some choused fault is performed by quality enhancement for one preliminary constructed solution. The data for evaluating the quality of the potential solution is brought from the fault simulation of the circuit under consideration. Such approach allows building the tests for the circuits of medium and high size. Also we give the results of the computer experiments on the ISCAS-89 benchmark circuits that show the effectiveness of the proposed approach.

Key words: sequential circuit, test generation, simulating annealing algorithm, fault simulation.

АЛГОРИТМ СИМУЛЯЦИИ ОТЖИГА ПОСТРОЕНИЯ ТЕСТОВ ЦИФРОВЫХ УСТРОЙСТВ

Д.Е. Иванов, Р. Зуауи

В статье предлагается новый двухуровневый алгоритм решения классической задачи генерации тестов для синхронных последовательностных схем. Он основан на оптимизационной стратегии симуляции отжига. Построение решения для некоторой выбранной неисправности ведётся путём улучшения свойств одного первоначального решения. Данные для построения оценки потенциального решения получаются на основе моделирования с неисправностями исследуемой схемы. Такой подход позволяет эффективно строить тесты для схем средней и большой размерности, что подтверждают результаты апробации на схемах из международного каталога ISCAS-89.

Ключевые слова: цифровая последовательностная схема, генерация тестов, симуляция отжига, моделирование с неисправностями.

Сведения об авторах

Иванов Дмитрий Евгеньевич

научная степень, учёное звание: кандидат технических наук, доцент;

должность: старший научный сотрудник отдела теории управляющих систем Института прикладной математики и механики НАН Украины, Донецк;
доцент кафедры АСУ, Донецкий национальный технический университет, Донецк;

научные интересы: генетические алгоритмы, техническая диагностика, генерация тестов, эволюционные оптимизационные стратегии, моделирование цифровых схем;

адрес: ул. Розы Люксембург, 74, Донецк, Украина, 83114;

e-mail: ivanov@iamm.ac.donetsk.ua;

телефоны: (062) 311-67-95 – рабочий;
(067) 281-2648 – мобильный;

Зуауи Рамзи

научная степень, учёное звание: нет;

должность: аспирант кафедры АСУ, Донецкий национальный технический университет, Донецк;

научные интересы: генерация тестов, алгоритмы симуляции отжига;

адрес: ул. Артёма, 58, Донецк, Украина, 83001;

e-mail: zouaoui_ram@hotmail.com;

телефоны: (062) 304-90-20 – рабочий.