УДК 681.518

ГЕНЕТИЧЕСКИЙ ПОДХОД ПРОВЕРКИ ЭКВИВАЛЕНТНОСТИ ПОСЛЕДОВАТЕЛЬНОСТНЫХ СХЕМ

Д.Е. Иванов

Under the life cycle of the development of the modern digital circuits a problem of verification of two circuits is arise. It is caused, for example, by using several optimization procedures. In this paper a genetic algorithm of solving this problem is proposed. Reported experimental results on the ISCAS-89 benchmarks confirm the efficient of the proposed algorithm.

При проектировании современных цифровых схем перед разработчиком часто возникает задача верификации эквивалентности двух схем, что связано с применением различных оптимизационных процедур. В данной статье предлагается алгоритм решения данной задачи, основанный на генетическом программировании. Приведены результаты экспериментов на схемах ISCAS-89, показывающие эффективность предложенного подхода

Під час проектування сучасних цифрових пристроїв перед розробником часто постає проблема верифікації еквівалентності двох схем Це пов'язано, перед усім, із використанням різноманітних оптимізуючих процедур. В даній статі запропоновано алгоритм рішення такої задачі, заснований на генетичному програмуванні. Наведено результати машинних експериментів на схемах ISCAS-89, що доводять ефективність запропонованого підходу.

Введение

На современном этапе средства автоматизированного проектирования позволяют обрабатывать как комбинационные схемы, содержащие десятки и сотни тысяч логических вентилей, так и последовательностные схемы, содержащие сотни элементов состояний (триггеров). Обязательными элементами таких систем являются программы трансляторы (например, между функциональным и логическим уровнями описания схем), а также программы оптимизации каких либо параметров схемы (потребляемой энергии, числа логических вентилей, удаление последовательностной избыточности и т.д.). Применение таких средств автоматизации заставляет разработчика ставить задачу верификации эквивалентности двух схем (например, схемы до и после процесса оптимизации) [1]. Для решения таких задач существуют точные алгоритмы, которые основаны, например, на преобразованиях булевого представления схемы [2]. Недостатками данных методов является

то, что при их работе со схемами большой размерности легко достигается переполнение памяти. Таким образом, будучи прерванными из-за проблемы переполнения памяти, точные алгоритмы вообще не дают никакого результата: ни об эквивалентности, ни о неэквивалентности заданных схем. Это заставляет разрабатывать новые нетрадиционные подходы к решению подобных задач. Одним из них является применение генетических алгоритмов [3]. Применение генетических алгоритмов, например, для построения тестовых входных последовательностей описано в [4].

Авторы также имеют опыт применения генетических алгоритмов к решению задач технической диагностики. В частности, такие алгоритмы использовались для построения тестовых входных последовательностей [5,6], а также инициализирующих последовательностей [7]. В данных алгоритмах операции производятся над входными двоичными последовательностями, состоящими из входных векторов, каждый из которых соответствует такту модельного времени. Для такого задания особей вводятся основные генетические операции: мутация и скрещивание. Вычисление оценочных функций при этом основывается на моделировании работы схемы (исправной или с неисправностями) и отображает активность схемы на заданной входной последовательности. Подход показывает приемлемые практические результаты. Поэтому авторы решили применить его для задачи верификации эквивалентности последовательностных схем.

Данная статья имеет следующую структуру. В первом разделе описана формальная модель синхронных последовательностных схем, используемая при исследовании. В следующем разделе описана общая структурная схема генетического алгоритма, применяемого для решения поставленной задачи. В третьем разделе подробно рассмотрено вычисление фитнесс-функции, основанное на моделирования. В четвёртом разделе описана программная реализация предложенного алгоритма, среда проведения машинных экспериментов, а также приведены их результаты. В заключении сделаны выводы и указаны направления дальнейших исследований.

1. Постановка задачи

В качестве модели в данной работе используются синхронные последовательностные схемы (рис.1). В данной модели схема представляется в виде комбинационного блока (в свою очередь состоящего из нескольких функциональных комбинационных блоков, КФБ) и блока памяти, который состоит из D-триггеров. Далее также будем использовать следующие обозначения: V - вектор входных сигналов; $\#Bx$ – число внешних входов схемы, размерность вектора V; Y - вектор выходных сигналов; $\#Вых$ – число внешних выходов схемы, размерность вектора Y; T - вектор состояний блока памяти; $\#Tp$ – число элементов

состояния (D-триггеров) схемы, размерность вектора T; $\#Эл$ - общее число логических вентилей в схеме.

Вектор V – упорядоченное множество двоичных значений, которое подаётся на вход цифровой схемы в определённый такт времени. Последовательность s_i заданной длины l_i - упорядоченное множество из l_i векторов, которые подаются на вход схемы в последовательные такты времени. Обозначение v_{ij} говорит, что мы рассматриваем в последовательности s_i вектор с номером j ($j = 0,...,l_i - 1$). Через $A(s)$ обозначим выходные реакции схемы A при подаче на её входы последовательности s. Далее везде предполагаем, что моделирование выполняется в 3-значном алфавите [8] и работа схемы начинается из полностью неопределённого состояния.

При моделировании используется преобразование синхронной последовательностной схемы в псевдокомбинационный эквивалент с дальнейшим его итеративным моделированием в последовательные такты времени. Для такого преобразования удаляют элементы состояний. Входы элементов состояний (вектор T_i на рис.1) при этом называются псевдовыходами, а их выходы (вектор T_{i-1} на рис.1) – псевдовходами.

При таком преобразовании работа блока памяти представляется следующим образом. В момент смены тактового импульса в устройстве синхронизации (для упрощения на рисунке не представлен) происходит подача на вход схемы новых входных значений v_i момента времени i; на псевдовходы подаются значения псевдовыходов схемы в предыдущий такт времени T_{i-1}, после чего путём моделирования комбинационной части схемы для такта времени i формируются выходные сигналы схемы Y_i и сигналы псевдовыходов T_i.

Задача верификации эквивалентности двух последовательностных схем формулируется следующим образом.

Определение 1. Пусть заданы две цифровые последовательностные схемы A_0 и A_1. Будем называть схему A_0 исправной, а схему A_1 - модифицированной (оптимизированной, неисправной). Схемы называются последовательностно эквивалентными (или просто эквивалентными), если при произвольном одинаковом начальном состоянии данных схем выходные реакции схем на произвольные входные последовательности s являются одинаковыми, т.е.

$$A_0 = A_1 \Leftrightarrow \forall s : \quad A_0(s) = A_1(s). \tag{1}$$

Как было сказано выше, построенные на основании деревьев обхода алгоритмы [2,3] не в состоянии работать со схемами большой размерности. Это связано с нехваткой памяти для стеков возвратов алгоритмов обхода деревьев, что обусловлено большим числом элементов состояний в проектируемых схемах.

Однако проектировщику не обязательно иметь утвердительный ответ на вопрос об эквивалентности схем. Ему достаточно также знать, что схемы не являются эквивалентными. Поэтому задачу можно переформулировать, ориентируясь на поиск контрпримера.

Определение 2. Если существует такая последовательность s_k, на которой при одинаковых начальных состояниях схемы A_0 и A_1 имеют различные выходные реакции, то данные схемы не являются эквивалентными.

$$A_0 \neq A_1 \ \textit{если} \ \exists\, s_k : A_0(s_k) \neq A_1(s_k) \qquad (2)$$

Именно поиск такой последовательности будет являться целью предлагаемого алгоритма. В данной постановке задача не рассматривалась в отечественной научной литературе.

Предполагается также, что проектировщику доступны значения сигналов при работе схемы в контрольных точках (КТ). В самом общем случае множество КТ совпадает с множеством логических элементов схемы, т.е. проектировщику известно полное поведение схемы в процессе моделирования. Однако ему не известно никакой дополнительной информации о внутренней функциональной структуре схемы.

2. Генетический алгоритм верификации эквивалентности последовательностных схем

Как уже отмечалось ранее, авторы неоднократно применяли различные модификации генетического алгоритма к задачам технической диагностики [5-7].

Генетический алгоритм это итеративный процесс построения новой популяции (поколения) из текущей. Алгоритм заканчивает работу при выполнении одного из условий:

- либо найдена последовательность s, выходные реакции на которую схем $A_0(s)$ и $A_1(s)$ различны (показана неэквивалентность схем);
- либо достигнуто одно из ограничивающих условий: время моделирования, предельное число поколений или предельное число поколений без улучшения оценочной функции.

В алгоритмах такого рода в качестве особи выступают входные последовательности,

для которых заранее не известна длина (рис.2а). Длина входного вектора соответствует числу входов исследуемых схем A_0 и A_1. Для каждой входной последовательности s_i ставится в соответствие оценочная функция $f_{оц}(s_i)$, которая показывает насколько «хорошо» данная последовательность решает поставленную задачу. Вычисление оценочных функций особей является наиболее трудоёмкой фазой алгоритма и именно от этого зависит быстродействие всего алгоритма. Подробно вычисление оценочной функции особей описано в следующем разделе. Популяцией является набор таких последовательностей (рис.2б).

Одна итерация генетического алгоритма состоит в построении новой популяции из текущей. При этом к особям текущей популяции применяют генетические операции: выбор особей, скрещивание и мутация особей. Операция выбора реализует выбор двух особей-родителей для последующей рекомбинации и мутации. Авторы применяют пропорциональный отбор, при котором вероятность особи быть выбранной в качестве родителя пропорциональна её фитнесс-функции. После этого к выбранным особям применяются операции скрещивания и мутации. Полученные таким образом особи-потомки попадают в промежуточную популяцию. Текущая популяция новой итерации алгоритма строится на основании лучших особей текущей и промежуточной популяций.

Подробнее генетические операции описаны в [5] и здесь не приводятся для краткости изложения.

Псевдокод генетического алгоритма построения входных последовательностей приведён ниже.

```
Генетический_алгоритм(Схема, РазмерПопуляции, ЧислоИтераций)
{
  // подготовка начальной популяции
  ПостроитьНачальнуюПопуляцию();
  ОценитьОсобей(НачальнаяПопуляция);
  НомерПопуляции=0;
  Пока( не_достигнут_критерий_остановки )
  {
    НоваяПозиция=0;
    Для( i=0 ; i<РазмерПопуляции ; i++)
    {
      ОперацияВыбора(РодительА, РодительБ);
      Если( rand() < Pc )
        ОперацияСкрещивания(РодительА, РодительБ, Потомок);
```

```
      Если( rand() < Pm )
        ОперацияМутации(Потомок);
      ДобавитьВПромежуточнуюПопуляцию(Потомок, НоваяПозиция);
      НоваяПозиция++;
    }
    ОценитьОсобей(ПромежуточнаяПопуляция);
    ПостроитьНовуюПопуляцию(РазмерПопуляции);
    НомерПопуляции++;
  }
  СоздатьОтчёт();
}
```

Поясним назначение некоторых функций. Процедура «ОценитьОсобей» вычисляет оценочную функцию всех особей переданной ей популяции. Функция «ОперацияВыбора» на основании оценочной функции для особей текущей популяции строит их фитнесс-фукнции и производит выбор двух особей-родителей, над которым далее будут проводиться генетические операции. Функция «ПостроитьНовуюПопуляцию» обрабатывает текущую и промежуточную популяции и на их основе строит текущую популяцию следующей итерации алгоритма. В эту популяцию попадают лучшие особи из текущей популяции предыдущего шага алгоритма и промежуточной популяции. После этого промежуточная популяция уничтожается. Процедура «СоздатьОтчёт» формирует выходные статистические данные, как для пользователя программы, так и для системы моделирования и генерации тестов АСМИД-Е [9].

3. Вычисление оценочной функции.

Идеей предлагаемого алгоритма является направить поиск в пространстве решений в ту сторону, в которой выше активность верифицируемых схем A_0 и A_1, а также выше различие этой активности на множестве контрольных точек. Данную информацию легко получить из программ логического моделирования. Таким образом, в генетических алгоритмах генерации различных типов тестовых последовательностей, вычисление фитнесс-функции основано нам моделировании работы исправной A_0 и модифицированной A_1 схем. Основываясь на вышесказанном, в качестве меры применяются следующие показатели: активность исправной или неисправной схемы на заданной входной последовательности; число вентилей, псевдовыходов и выходов в схеме, различных в исправной и неисправной схемах. В нашем случае роль «неисправной» схемы выполняет схема, которая подверглась обработке

(оптимизации) некоторым транслятором. Поэтому для построения фитнесс-функции выберем меру отличия значения сигналов в исправной и оптимизированной схемах. Основываясь на данном факте, введём три параметра, на основании которых будет вычисляться численное значение фитнесс-функции:

- n1 – число различных значений на выходах схемы. Данный параметр является определяющим, поскольку наличие даже одного расхождения на внешних входах двух схем, говорит о том, что различающая последовательность построена. Поэтому данный параметр в фитнесс-функцию должен входить с таким коэффициентом, чтобы при наличии хотя бы одного различия, значение фитнесс-функции нивелировало другие параметры и говорило о решении задачи.
- n2 – число различных псевдовыходов схемы. В случае, когда различные значения в двух проверяемых схемах, ещё не достигли выходов, решающим является распространения таких отличий на псевдовыходы (триггеры) схемы. Наличие триггеров с различными значениями на определённом такте времени позволяет с большей вероятностью получить различия на внешних выходах схемы на следующем такте работы.
- n3 – число контрольных точек двух схем с различными значениями сигналов. В случае, когда различие в поведении двух схем не достигло ни внешнего выхода, ни псевдовыхода схемы, необходимо строить последовательности, для которых является важным различие сигналов внутри комбинационных блоков. Этот параметр важен либо когда ещё ни одна из пар триггеров в исправной и неисправной схемах не получили различные значения, либо когда такие пары есть, но этого не достаточно для распространения различных значения до внешних выходов схемы. Данный параметр в фитнесс функции должен иметь наименьшее значение.

Таким образом, в общем виде значение фитнесс-функции имеет вид:

$$f(A_0, A_1, s_i) = \sum_{j=1}^{\text{длина } s_i} f(A_1, A_0, s_{ij}) = \sum_{j=1}^{\text{длина } s_i} (c_1 * n_1 + c_2 * n_2 + c_3 * n_3), \quad (3)$$

где:

$c_1 - c_3$ - нормирующие константы;

$n1 = \sum_{g \in G} различие(g, A_0, A_1, s_{ij})$, G – множество всех внешних выходов двух схем;

$n2 = \sum_{g \in G1} различие(g, A_0, A_1, s_{ij})$, G1 – множество псевдовыходов двух схем;

$n3 = \sum_{g \in G2} различие(g, A_0, A_1, s_{ij})$, G2 – множество всех контрольных точек схемы. В нашем случае мы предполагаем, что проектировщику известно поведение всей схемы, поэтому множество $G2$ совпадает с множеством всех вентилей схемы.

Функция «различие(g)» вычисляется на основе моделирования и определяется следующим образом:

$$различие(g, A_0, A_1, s_{ij}) = \begin{cases} 1, \text{если значения сигналов на выходе блока } g \text{ различны в двух} \\ \text{схемах } A_0 \text{ и } A_1 \text{ при моделировании на входном наборе } s_{ij}; \\ 0, \text{в противном случае.} \end{cases} \quad (4)$$

Псевдокод процедуры вычисления оценочной функции одиночной последовательности приведён ниже.

```
Вычисление_оценочной_функции(Схема, ВходнаяПоследовательность, Длина)
{
  Фитнесс=0;
  для(i=0;i<длина;i++)
  {
    SV0=МоделированиеРаботыИсправнойСхемы(набор i)
    SV1=МоделированиРаботыМодиф.Схемы(набор i)
    n1=ЧислоРазличныхВыходов(SV0,SV1);
    n2=ЧислоРазличныхПсевдовыходов(SV0,SV1);
    n3=ЧислоРазличныхВентилей;
    Фитнесс=Фитнесс+c1*n1+c2*n2+ c3*n3;
  }
}
```

Здесь SV0 и SV0 массивы, содержащие значения сигналов на всех контрольных линиях двух сравниваемых схем A_0 и A_1[9].

4. Программная реализация и экспериментальные данные.

Для проведения машинных экспериментов авторы реализовывали «почти эквивалентные» схемы основываясь на следующих соображениях. Обработка описания цифровой схемы (переход между уровнями описания: вентильный<->функциональный и

т.д.) происходит автоматизировано с помощью соответствующих трансляторов. При этом преобразованию подвергаются некоторые завершённые функциональные блоки. Вероятность возникновения ошибки при таких преобразованиях достаточна мала, но не является нулевой. При этом будем предполагать, что она возникнет только в одном из логических блоков, и при моделировании будет проявляться (не всегда) на выходе (выходах) данного блока. При описании схемы на вентильном уровне минимальным элементом описания является логический вентиль. А функционирование одного логического вентиля определяется его типом. Изменив тип логического вентиля, будет изменено его функциональное свойство. Таким образом, для построения «почти эквивалентной схемы» будем изменять в таблицах описания тип одного случайно выбранного вентиля. Полученная таким образом схема и оригинальная схема будут верифицироваться на эквивалентность.

Программная реализация производилась в среде программирования C++ Builder 6. Общее число строк кода, реализующих функциональную часть алгоритма без процедур моделирования, составило около 1200 строк.

Для реализации «почти эквивалентной» модифицированной схемы создавался второй массив типов элементов R_TYPES [9], в котором случайным образом изменялся тип одного вентиля (например OR->NOR). Дважды (для исправной и «эквивалентной» схем) вызывалась процедура моделирования, для которой в параметрах подменялся один единственный массив R_TYPES. Остальные массивы в принятом подходе описания двух схем одинаковы и не дублировались. Для моделирования работы цифровых последовательностных схем использовался разработанный авторами ранее алгоритм событийного моделирования [10].
В качестве экспериментальной платформы использовался персональный компьютер со следующими характеристиками: процессор Intel CoreQuad с частотой 2.4Ггц, объем оперативной памяти 2Гбайта.

При проведении машинных экспериментов использовались следующие значения эвристических констант:

- $c_1 = 1$, если различие в поведении двух верифицируемых схем достигнет внешнего выхода, то значение оценочной функции превысит единицу, что будет сигнализировать, что найдена требуемая входная последовательность;

- $c_2 = \frac{1}{\#Вых \cdot \#Тр}$, при таком выборе константы различие на всех линиях элементов состояний будет приравниваться к различию на одном внешнем выходе схемы, показывая высокую вероятность распространить данное отличие на какой либо внешний выход схемы в следующем такте времени;

- $c_3 = \frac{1}{\#Вых \cdot \#Эл}$, аналогично, различие на выходах всех вентилей схемы приравнивается к различию на одном внешнем выходе.

Результаты машинных экспериментов для некоторых больших схем каталога ISCAS-89 приведены в табл.1. Поскольку предлагаемый алгоритм не является точным, то объясним смысловое различие колонок «Не различено схем» и «Ответ под вопросом». Данные в столбце «Не различено схем» показывают число экспериментов, в которых достигнут предел числа итераций, однако не найдено различающей последовательности, а оценочная функция всех последовательностей популяции равна нулю. Данные в столбце «Ответ под вопросом» показывают число машинных экспериментов, в которых достигнут предел построения популяций, однако оценочная функция отлична от нуля. Таким образом, этот столбец соответствует случаю, когда функционирование схем различно, но данное различие не удалось распространить на внешние выходы схемы. При этом остаётся вероятность того, что при углублении поиска (увеличении числа предельно допустимых итераций алгоритма) зафиксированное различие в поведении двух схем удастся распространить на внешние выходы схем, т.е. построить различающую их последовательность.

Приведённые в табл.1 числовые данные показывают эффективность предложенного алгоритма: число экспериментов, в которых удалось построить различающую функцию, составило 94.7%. Число экспериментов, в которых различие функционирования не проявилось даже внутри схемы, составило 2%.

Таблица 1. Экспериментальные данные работы алгоритма верификации эквивалентности.

Имя схемы	Число вентилей / триггеров схемы	Эксперименты			
		Всего проведено	Различено схем	Не различено схем	Ответ под вопросом
s3271	1731 / 116	25	25	0	0
s3330	2037 / 131	25	21	2	2
s3384	1940 / 183	25	24	1	0
s4863	2514 / 104	25	25	0	0
s5378	3045 / 179	25	22	0	3
s6669	3460 / 239	25	25	0	0
Всего:		150	142 (94.7%)	3 (2%)	5 (3.3%)

Выводы

В данной статье предложен генетический подход к решению задачи проверки эквивалентности двух синхронных последовательностных схем. Авторами разработан алгоритм, который основан на предыдущем опыте построения входных тестовых последовательностей. Проведена программная реализация данного алгоритма и ряд машинных экспериментов. Апробация программной реализации алгоритма на схемах из международного каталога ISCAS-89 показывает его высокую эффективность как в терминах качества построения различающих последовательностей, так и в терминах времени их построения.

В качестве дальнейших исследований данной проблемы можно отметить следующее. Вычисление оценочной функции (раздел №) строящейся последовательности требует моделирования работы двух сравниваемых схем. В терминах параллельного программирования две такие процедуры моделирования не имеют пересечений и считаются независимыми. Таким образом можно реализовать их параллельное вычисление при наличии соответствующей аппаратной поддержки. Этот подход легко достижим на современных многоядерных процессорах. Поскольку в данном алгоритме вычисление оценочной функции забирает большую часть времени, то переход от последовательного моделирования работы двух схем к параллельному должен существенно уменьшить общее время работы всего алгоритма. Построение модификации алгоритма для многоядерных процессоров и проверка эффективности такого подхода будет являться ближайшей целью авторов.

Литература

1. S.-Y. Huang, K.-T. Chen, “*Formal Equivalence Checking and Design Debugging”*, Kluwer Academic Publishers.- Boston.- 1998.- 229p.
2. Ghosh, A., S. Devadas and A. R. Newton, ``*Sequential Logic Testing and Verification"*, Kluwer Academic Publishers.- 1992.- 214p.
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. 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. Skobtsov Y.A., El-Khatib, Ivanov D.E. *Distributed Genetic Algorithm of Test Generation For*

Digital Circuits // Proceedings of the 10th Biennial Baltic Electronics Conference.-Tallinn Technical University,2006.-p.281-284. (0.4 д.а.)

7. 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.
8. Барашко А.С., Скобцов Ю.А., Сперанский Д.В. *Моделирование и тестирование дискретных устройств.* – Киев:Наукова думка, 1992. – 288 с.
9. Скобцов Ю.А., Иванов Д.Е. *Автоматизированная система моделирования и генерации тестов АСМИД-Е* // Техническая диагностика и неразрушающий контроль. - 2000. - №2. - С.54-59.
10. Иванов Д.Е., Скобцов Ю.А. *Параллельное моделирование неисправностей для последовательностных схем* // Искусственный интеллект. – 1999. - №1. – С.44-50.