ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ В ИНТЕЛЛЕКТУАЛЬНЫХ САПР

101—109

Скобцов Ю.А.
Донецкий государственный технический университет, кафедра АСУ,
E-mail:skobtsov@kita.dgtu.donetsk.ua

Рассматривается применение генетических алгоритмов в системах автоматизированного проектирования СБИС. Описано применение генетических алгоритмов к построению тестов логических схем и к задачам конструкторско-технического проектирования.

Genetic algorithms application in computer aided design VLSI is considered. There is described the application of genetic algorithms to logical circuit test generation and VLSI physical design automation.

Введение. В основе интеллектуальных САПР (ИСАПР) лежит использование методов искусственного интеллекта. ИСАПР является эффективным средством проектирования и производства современных компьютерных систем. Под ИСАПР подразумевается организационно-техническая система, состоящая из интеллектуального комплекса средств автоматизации проектирования, взаимодействующая с пользователями и компьютерными сетями и выполняющая автоматическое проектирование. ИСАПР состоит из совокупности средств и компонентов математического, информационного, программного, технического, лингвистического, методического и организационного обеспечения.

Общий цикл проектирования современной элементной базы компьютерных систем на основе сверхбольших интегральных схем (СБИС) включает следующие основные этапы :системная спецификация, функциональное проектирование, логическое проектирование, проектирование цепей, конструкторское проектирование, изготовление, упаковка и тестирование. В дальнейшем мы, в основном, рассмотрим этапы логического и конструкторского проектирования. Отметим также что около 50% стоимости проектирования и изготовления СБИС составляют затраты на их диагностирование и тестирова-

ние. Поэтому задачи диагностики и тестирования СБИС имеют большое значение.

Для повышения уровня "интеллекта" ИСАПР, расширения ее возможностей и повышения эффективности применяются различные методы искусственного интеллекта. Одним из самых перспективных направлений является использование генетических алгоритмов (ГА) и эволюционного моделирования.

Генетические алгоритмы. Генетические алгоритмы были разработаны Холландом [1] и далее хорошо зарекомендовали себя при решении задач численной и комбинаторной оптимизации. Генетические алгоритмы используют случайный направленный поиск для построения (суб)оптимального решения данной проблемы. При этом направленный случайный поиск моделирует процесс естественной эволюции. Каждое решение задачи (особь) представляется хромосомой – стрингом элементов (генов). Классический "простой" генетический алгоритм [1] использует двоичные стринги –строки из двоичных элементов 0,1 (например, 0011101), что делает его привлекательным для решения задач генерации проверяющих тестов логических схем , где решение задачи представляется в виде двоичных наборов или их последовательностей. На множестве решений определяется целевая (fitness) функция (ЦФ), которая позволяет оценить близость каждой особи к оптимальному решению. Множество особей (решений) образует популяцию которая развивается (эволюционирует) от одного поколения к другому.

Механизм простого ГА достаточно прост. Он копирует стринги и переставляет их части. Предварительно ГА случайно строит начальную популяцию, к которой далее применяются несколько простых операций и генерируются новые популяции. Простой ГА использует три основных оператора: репродукция, кроссинговер, мутация. При репродукции хромосомы копируются согласно их значениям ЦФ. Копирование лучших хромосом с большими значениями ЦФ определяет большую вероятность их попадания в следующую генерацию. Оператор репродукции реализует принцип "выживания сильнейших" по Дарвину.

Оператор кроссинговера (ОК) обычно выполняется в три этапа.

1). Два стринга (хромосомы, особи) $A= a_1a_2 \ldots a_n$ и $B=b_1b_2\ldots b_n$ выбираются случайно из текущей популяции после репродукции.

2). Выбирается также случайно точка ОК k (1=<k<n).

3) Хромосомы А и В обмениваются частями после k-й позиции и производят два новых стринга $A' = a_1a_2 \ldots a_kb_{k+1}\ldots b_n$ и $B'=b_1b_2\ldots b_k b_{k+1}\ldots b_n$.

Оператор мутации (ОМ) случайным образом (с небольшой вероятностью) изменяет ген (элемент) стринга.

Эти три простых оператора образуют удивительно мощный поисковый механизм.

ГА отличаются от других оптимизационных и поисковых процедур согласно следующими чертами: работают, в основном, не параметрами, а с закодированным множеством параметров; выполняют поиск из популяции точек, а не из единственной точки; для оценки информации используют саму целевую функцию, а не ее различные приращения; используют не детерминированные, а вероятностные методы.

При решении задачи ГА работает до тех пор пока не будет достигнута верхняя граница числа итераций или на некоторой генерации будет получено решение приемлемого качества или возникнет преждевременная сходимость в результате попадания в локальный минимум. В отличие от других методов оптимизации ГА обычно ищут оптимум в различных областях пространства решений одновременно и поэтому более приспособлены к поиску новых перспективных областей за счет объединения квазиоптимальных решений их разных популяций. На основе простого ГА можно разработать практически любую программу генетического поиска для реализации различных прикладных задач с использованием элементов адаптации.

Далее были предложены многочисленные обобщения как основных генетических операторов ОК, ОМ ,так и самого ГА [5]. При решении конкретной задачи прежде всего надо разработать форму представления (кодирование) решения (особи) и определить на ней основные генетические операторы ОК, ОМ. Далее мы на примере задач ИСАПР рассмотрим эти проблемы.

Применение ГА при логическом проектировании и генерации проверяющих тестов. Задача генерации тестов для логических схем является одной из самых сложных и трудоемких. Ее можно сформулировать следующим образом. Для данной логической схемы, представленной в виде соединений логических вентилей и триггеров

(или описанной на специализированном языке типа VHDL) необходимо построить входную двоичную последовательность, проверяющую достаточно высокий процент неисправностей. При этом обычно рассматриваются модельные одиночные константные неисправности. Использование ГА при решении этой задачи является естественным развитием псевдослучайных методов генерации тестов [2] . При применении ГА для решения этой задачи в простейшем случае в качестве особей рассматриваются отдельные тестовые двоичные наборы (генами являются значения сигналов на входах схемы) [3]. Популяцией является тестовая последовательность. Тогда используются стандартные операторы ОК и ОМ на этих двоичных стрингах.

Поскольку целью генерации тестов является построение последовательности, на которой максимально отличаются значения сигналов в исправной и неисправной схемах то 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.

Использование отдельных тестовых наборов в качестве особей ГА, как правило, эффективно только на начальном этапе генерации тестов. Далее, особенно для последовательностных схем, целесообразно в качестве особи брать всю тестовую последовательность, длина которой заранее не известна и не ограничена (Рис. 1а.) [4]. Она состоит из векторов, каждый бит которых соответствует входу схемы. В свою очередь популяция состоит из заранее заданного числа особей (Рис 1б.).

Фитнесс-функция последовательности для фиксированной неисправности вычисляется из фитнесс- функций всех её векторов по формуле:

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

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<=1, благодаря которой предпочтение отдаётся более коротким последовательностям.

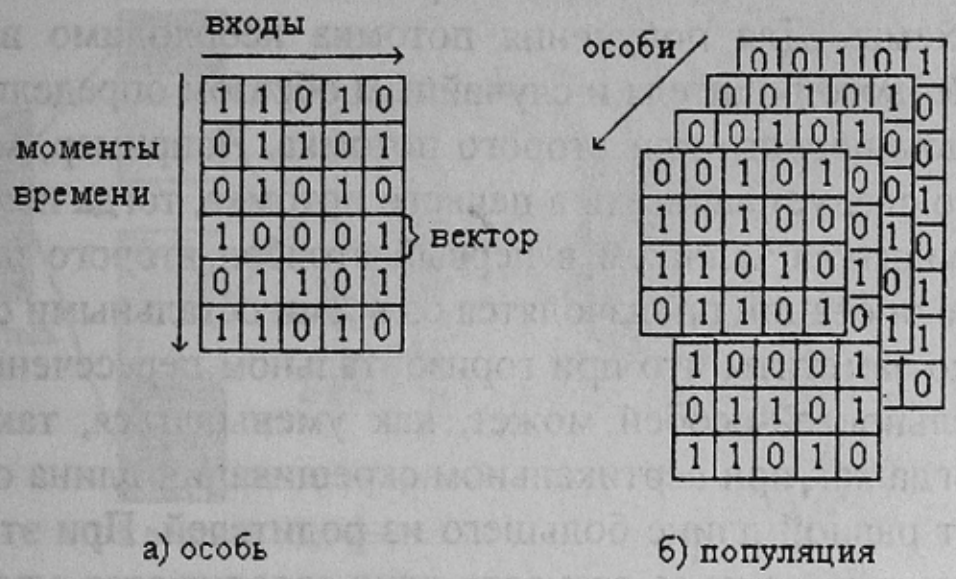

Рисунок 1. Кодирование особей и популяций.

Здесь фитнесс-функция одного вектора из последовательности имеет вид:

$$h(v,f) = f_1(v,f) + c_1 * f_2(v,f) \qquad (2)$$

где 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 вентилей 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.

В этом случае используется два вида ОК, выбор между которыми производится случайно: горизонтальное и вертикальное [4].

При горизонтальном скрещивании случайным образом в последовательностях-родителях выбираются точки разреза x_1 и x_2. Первый потомок получается соединением векторов первого родителя от начала последовательности до точки деления x_1 и второго родителя от точки деления x_2 до конца последовательности (Рис. 2а.). Второй потомок получается аналогично при перемене мест первого и второго родителя;

Вертикальное скрещивание производится не по векторам, а по входам схемы. Для получения потомка необходимо взять первый столбец первого родителя и случайным образом определить, будет ли он столбцом первого или второго потомка. Например, мы определили, что его следует записать в первого потомка, тогда первый столбец второго родителя запишем в первый столбец второго потомка (Рис. 2б.). Те же операции производятся со всеми остальными столбцами.

Легко заметить, что при горизонтальном пересечении длина последовательностей-особей может, как уменьшаться, так и увеличиваться, тогда как при вертикальном скрещивании длина обоих потомков станет равной длине большего из родителей. При этом все «пустоты», получаемые из-за разности длин заполняются случайными битами.

Механизм мутации обеспечивает появление новых генов в особях. Он применяется с заранее заданной вероятностью P_m к выходу операции скрещивания. В алгоритме применяется три вида мутации, выбор между которыми происходит случайно: 1) мутация одиночного бита (каждый бит в последовательностях-потомках может изменить своё значение с вероятностью 0.5%), 2) добавление вектора (в случайную позицию в последовательности вставляется случайным образом сгенерированный вектор, что позволяет вводить и рассматривать тестовые последовательности с большей длиной), 3) удаление вектора (из случайно выбранной позиции удаляется вектор, если он был несущественным, то оценочная функция всей последовательности не должна уменьшится).

Применение ГА для построения проверяющих тестов показало их высокую эффективность по сравнению с традиционными методами генерации тестов, в особенности, для последовательностных схем [4]. Самым эффективным в настоящее время является совместное применение и чередование детерминированных и генетических методов генерации тестов. Возможно и многоуровневое применение ГА в генерации тестов. При этом большая схема разбивается на подсхемы, и строятся тесты для подсхем с использованием ГА. Далее эти тесты объединяются в тестовую последовательность для всей схемы также с применением ГА.

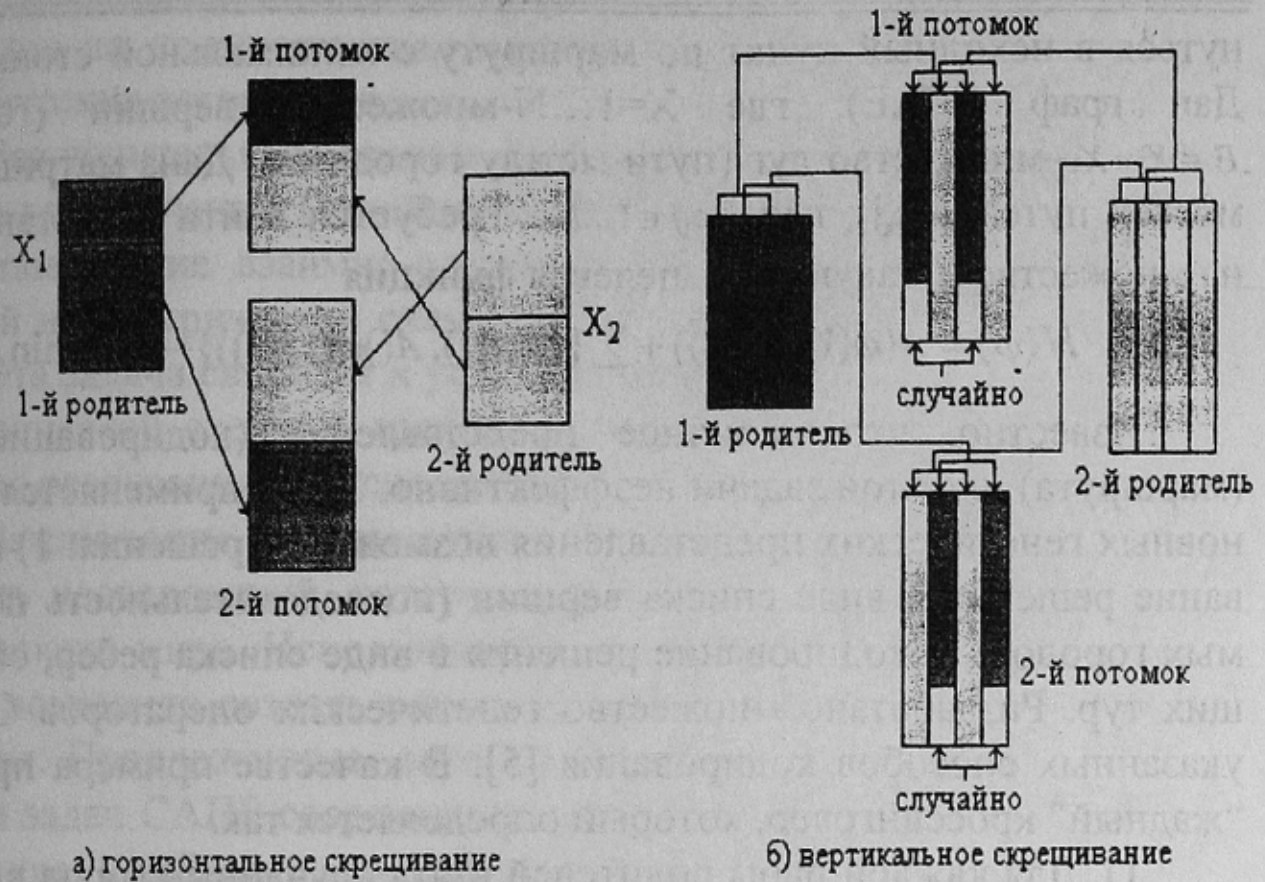

Рисунок 2. Операции скрещивания

При таком подходе для типовых подсхем (например, АЛУ, ОЗУ и т.п.) можно использовать стандартные тесты в качестве отдельных особей. Аналогично при построении проверяющих экспериментов с автоматами на первом уровне целесообразно с использованием ГА строить специальные идентифицирующие входные последовательности (такие как синхронизирующая, диагностирующая, установочная, уникальная и т.п. [2]). Далее на втором уровне, используя в качестве особей эти последовательности и с учетом знания структуры эксперимента можно строить проверяющие эксперименты.

Применение ГА на этапе конструкторско-технического проектирования. На данном этапе ГА применяются при решении различных задач комбинаторной оптимизации таких как размещение, упаковка, компоновка элементов, трассировка и т.п. Базовой задачей проблем комбинаторной оптимизации является классическая NP-полная задача коммивояжера. Она используется при проектировании разводки коммутаций, разработке архитектуры вычислительных сетей и т.п. Ее постановка следующая. Коммивояжеру необходимо посетить N городов, не заезжая в один и тот же город дважды, и вер-

нуться в исходный пункт по маршруту с минимальной стоимостью. Дан граф G(X,E), где X=1…N-множество вершин (городов), $E \in X \times X$ –множество дуг (пути между городами). Дана матрица стоимостей путей A(i,j), где $i < j \in 1...N$. Требуется найти перестановку φ на множестве X такую, что целевая функция

$$H(\varphi) = A(\varphi(1), \varphi(N)) + \sum \{A(\varphi(i), A()\varphi(i+1))\} \longrightarrow \min .$$

Известно, что двоичное представление (кодирование) тура (маршрута) для этой задачи неэффективно. Здесь применяется два основных генетических представления возможного решения: 1) кодирование решения в виде списка вершин (последовательность посещаемых городов),2) кодирование решения в виде списка ребер, образующих тур. Разработано множество генетических операторов ОК для указанных способов кодирования [5]. В качестве примера приведем "жадный" кроссинговер, который определяется так.

1). Для каждой пары родителей взять случайный город в качестве точки старта и взять его в качестве текущего города.

2). Сравнить два ребра (пути), ведущих из текущего города и выбрать из них кратчайший.

3). Если выбранный таким образом город уже посещался, то взять случайный город из множества еще непосещенных. Присвоить полученный таким образом город текущему.

4). Повторять п.2 и 3, пока все города не будут посещены.

Оператор ОМ можно, например, определить как случайную перестановку двух городов в туре.

При решении задачи трассировки часто применяют алгоритмы построения кратчайших связывающих деревьев Прима и Штейнера. При этом обычно ставится задача соединения заданных точек (контактов) кратчайшим образом (сумма расстояний между точками должна быть минимальной или стремиться к ней). При решении этих задач применяются методы такого же типа как и при решении задачи коммивояжера.

Задачу размещения элементов СБИС можно определить как задачу упаковки максимального числа элементов различного размера в определенное число линеек одинаковой длины. Это также задача комбинаторной оптимизации, связанная с разделением множества

объектов на подмножества, которая успешно решается с помощью генетических алгоритмов.

Заключительным этапом конструкторского проектирования является верификация. Одной из важнейших задач верификации является установление взаимно-однозначного соответствия между структурной и электрической схемой СБИС и ее топологической реализации. Эта задача сводится к установления изоморфизма между графом электрической схемы и графом ее топологии и также имеет эффективную реализацию на основе генетических алгоритмов.

Заключение. Генетические алгоритмы – новая фундаментальная область исследований, которая моделирует механизмы развития органического мира. Использование концепций искусственного интеллекта позволило создать описание новых моделей процесса проектирования. Предложенные методы эффективно используются при решении задач САПР современных компьютерных систем, имеющих, в основном, комбинаторно-логический характер.

Список источников.

1. Holland J.P. Adaptation in Natural Artificial systems.University of Michigan Press,Ann Arbor,1975.

2. А.С. Барашко, Ю.А. Скобцов, Д.В. Сперанский, «Моделирование и тестирование дискретных устройств», Киев, Наукова Думка, 1992

3. E. M. Rudnick, J. G. Holm, D. G. Saab, J. H. Patel, "Application of Simple Genetic Algorithm to Sequential Circuit Test Generation", Proc. European Design & Test Conf., 1994, pp. 40-45.

4. 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.

5. Курейчик В.М. Генетические алгоритмы. Состояние. Проблемы. Перспективы//Известия академии наук.Теория и системы управления.1999,№1.-с.144-160.