ПОСТРОЕНИЕ ТЕСТОВ ЛОГИЧЕСКИХ СХЕМ НА ОСНОВЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

А.Е. Люлькин

Белгосуниверситет, механико-математический факультет, Независимости 4, 220030 Минск, Беларусь
lulkin@bsu.by

Несмотря на то, что к настоящему времени предложены различные методы и алгоритмы построения тестов для логических схем, существуют схемы, для которых не удается получить удовлетворительное решение за приемлемое время. Это обусловлено высокой трудоемкостью переборных задач, которые решаются в процессе построения тестов. В связи с этим представляет интерес применение идей и методов из теории генетических алгоритмов (ГА) для решения рассматриваемой задачи в тех случаях, когда это не удается сделать известными методами.

В настоящей работе предлагается один из вариантов ГА, который ориентирован на построение тестов для комбинационных логических схем (схем без памяти). Обсуждаются результаты экспериментального исследования программной реализации алгоритма на ряде схем различной сложности.

Пусть $C = \{C_1, C_2, \ldots, C_r\}$ - разбиение множества входов схемы $\{1, 2, \ldots, n\}$ на подмножества $C_j = \{j_1, j_2, \ldots, j_{u_i}\}$, где $j_1, j_2, \ldots, j_{u_i} \in \{1, 2, \ldots, n\}$. Пусть также N_1 — максимальное число случайных наборов, которые используются для формирования начального отрезка теста T (исходная популяция); N_2 - максимальное число входных наборов, генерируемых в процессе одной итерации на основе ГА; N_3 — максимальное число итераций.

1. С использованием моделирования неисправностей на случайных входных наборах строится некоторый начальный тест T (исходная популяция). $I := 1$.

2. Наборы из теста T упорядочиваются в порядке убывания числа обнаруживаемых ими неисправностей. Формируется множество наборов T_1. В T_1 включаются наборы с лучшим качеством. При этом $|T_1| = [|T|/3]$. $K := 1$.

3. Генерируется случайное число $p \in \{0, 1\}$. Если $p = 0$, то наборы t_1 и t_2, которые будут использоваться для построения очередного тестового набора t на основе ГА, выбираются из T_1, иначе — из T.

4. Случайным образом упорядочиваются подмножества $C_1, C_2, \ldots, C_r$ из C. В результате получим последовательность $C^u = (C_{i_1}, C_{i_2}, \ldots, C_{i_r})$; $i_1, i_2, \ldots, i_r \in \{1, 2, \ldots, r\}$.

5. Строим входной набор t на основе выполнения операции смешивания наборов t_1 и t_2 и операции мутации. $v := 1$.

6. Генерируется случайное число $p \in \{0,1\}$. Если $p = 0$, то в набор t включаем значения входов C_{i_v}, взятые из набора t_1, иначе — из набора t_2. Если $v = r$, то переход к п.7, иначе $v := v + 1$ и переход к началу п.6.

7. Выполняется операция мутации для полученного набора t. Генерируются случайные числа $p_1, p_2, \ldots, p_n \in \{1, 2 \ldots, n\}$. Если $p_j = 1$, то j-й разряд в t инвертируется.

8. Для набора t находится множество проверяемых им неисправностей и подсчитывается их число k_t. Если $k_t > 0$, то набор t включается в множество наборов T_I, генерируемых на итерации I для включения в тест (перед началом итерации $T_I = \varnothing$). Если список непроверенных неисправностей является пустым, то $T := T \cup T_I$ и выполнение алгоритма завершается. Иначе, если $K = N_2$, то $T := T + T_I$ и переход к п.9; иначе $K := K + 1$ и переход к п.3.

9. Если $I = N_3$, то тест T построен и выполнение алгоритма завершается; иначе $I := I + 1$ и переход к п. 2.