РАЗРАБОТКА И ПРИМЕНЕНИЕ ПРОГРАММНЫХ СРЕДСТВ ДЛЯ ОБУЧЕНИЯ СТУДЕНТОВ СПЕЦИАЛИЗАЦИИ «МАТЕМАТИЧЕСКАЯ ЭЛЕКТРОНИКА»

А. Е. Люлькин

Белорусский государственный университет
Минск, Беларусь
E-mail: lulkin@bsu.by

Рассматриваются оригинальные программные средства, которые используются при подготовке студентов специализации «Математическая электроника» Белгосуниверситета в области моделирования и тестирования логических схем. Приводится описание генетического алгоритма для построения теста, реализованного и включенного в состав программных средств.

Ключевые слова: логическое моделирование, тесты логического контроля, генетические алгоритмы.

С целью повышения качества подготовки студентов специализации «Математическая электроника» механико-математического факультета Белгосуниверситета в рамках международной программы REASON разработан спецкурс «Тестирование и тестопригодное проектирование БИС/СБИС». Спецкурс включает как лекционный материал, так и лабораторные занятия, направленные на получение практических навыков логического анализа цифровых схем с применением программных средств.

Известно, что разработка тестов логического контроля БИС/СБИС является одной из наиболее сложных и трудоемких задач, которые решаются в процессе их проектирования. К настоящему времени разработаны различные методы решения данной задачи, которые отличаются типом дискретных устройств (ДУ), для которых они предназначены, используемыми математическими моделями, классом рассматриваемых неисправностей, качеством и трудоемкостью получаемого решения, возможностью оптимизации построенного теста и др. В то же время некоторые методы являются конкурирующими, что требует их тщательного экспериментального исследования с целью выделения областей наиболее эффективного применения. Перечисленные выше вопросы рассматриваются в теоретической части курса.

Однако знание методов построения тестов еще не является достаточным условием для эффективного автоматизированного построения тестов с помощью известных программных систем. Это обусловлено сложностью объекта диагностирования, которая определяется многими параметрами, и проблемой обоснованного выбора самих определяющих параметров, а также громоздкостью методов построения тестов, как правило, требующих решения многих задач. В целях приобретения студентами практических навыков проектирования тестов были разработаны две подсистемы для автоматизированного построения тестов:

- подсистема анализа и построения тестов для тестопригодных схем;
- подсистема анализа и построения тестов для функционально-переключательных КМОП-схем общего вида.

Указанные программные средства реализованы в виде WINDOWS-приложений и фактически решают те же задачи, что и известные фирменные средства, но отличаются от них простотой использования, большими возможностями по отображению (раскрытию) процесса построения теста, наличием специальных средств для исследования алгоритмов решения задач (в том числе возможностью варьировать значениями параметров, с помощью которых осуществляется управление алгоритмами; генераторы псевдослучайных схем с заданными параметрами, позволяющие выполнить статистическое исследование алгоритмов). В рамках данных средств реализованы как известные методы построения тестов, так и новые подходы в области автоматизированного построения тестов: генетические алгоритмы; моделирование неисправностей из расширенного класса в КМОП-схемах и др.

Методы решения основных задач. Построение проверяющего теста для комбинационной логической схемы может быть выполнено вероятностным и регулярным методами.

Вероятностный метод состоит в следующем:

1) генерируется очередной псевдослучайный входной набор;
2) вычисляется множество неисправностей, проверяемых набором;
3) набор включается в тест, если он проверяет некоторые неисправности, которые не проверяются предшествующими наборами.

Построение теста завершается, если очередная группа испытываемых входных наборов, состоящая из M_1 наборов, проверяет меньше, чем M_2 новых неисправностей. Параметры M_1 и M_2 определяются опытным путем и от их значений зависит полнота построенного теста и требуемые затраты времени. Для определения проверяющих возможностей входных наборов используется параллельное моделирование неисправностей в двоичном алфавите.

Регулярный метод предназначен для построения проверяющих наборов для неисправностей, оставшихся непроверенными после применения вероятностного метода. Метод представляет собой модификацию метода существенных путей со структурным подходом к вычислению условий образования существенного пути (условий транспортировки неисправности к выходным полюсам схемы) и их обеспечению. К особенностям модификации метода относятся: возможность обработки функционально-сложных элементов, функции которых заданы системой дизъюнктивных нормальных форм, представленной в матричной форме; сочетание аналитического решения задач поиска условий проявления неисправности (вычисление *D*-кубов неисправности), транспортировки ее через логические элементы (вычисление *D*-кубов), расширения фиксации для отдельных логических элементов со структурным подходом при решении данных задач для схемы в целом.

Анализ и построение тестов для функционально-переключательных КМОП-схем выполняется с учетом расширенного класса неисправностей.

Построение теста выполняется вероятностным методом, который имеет ту же общую схему, что и для комбинационных схем. Для вычисления множеств неисправностей, проверяемых входными наборами, используется параллельное моделирование неисправностей. При этом моделирование функциональных элементов выполняется в троичном алфавите, а фрагментов, заданных на переключательном уровне, – в семизначном алфавите. Применение семизначного алфавита позволяет значительно повысить точность моделирования на переключательном уровне и учесть особенности проверки неисправностей из расширенного класса. В то же время моделирование функциональных элементов в троич-

ном алфавите значительно сокращает трудоемкость вычислительного процесса без потери точности моделирования. Реализовано асинхронное моделирование с учетом единичных задержек элементов.

Учитывая высокую трудоемкость построения тестов, а также то, что известные методы не гарантируют удовлетворительное решение задачи тестирования для произвольных схем, разработан модуль, реализующий построение тестов на основе генетических алгоритмов. Генетические алгоритмы представляют собой одно из направлений общей тенденции использования естественных аналогов при создании моделей, технологий, методик, алгоритмов для решения тех или иных задач. Во многих случаях использование естественных аналогов дает положительные результаты. Это объясняется тем, что аналог, взятый из природы, совершенствовался в течение многих лет эволюции и имеет на данный момент самую совершенную в своем роде структуру.

Генетические алгоритмы – это мощная стратегия выхода из локальных оптимумов. Она заключается в параллельной обработке множества альтернативных решений, концентрируя поиск на наиболее перспективных из них.

Рассмотрим особенности разработанного и реализованного генетического алгоритма (ГА) для построения тестов.

Пусть $C = \{C_1, C_2, ..., C_r\}$ – разбиение множества входов схемы $\{1, 2, ..., n\}$ на подмножества $C_j = \{j_1, j_2, ..., j_{u_i}\}$, где $j_1, j_2, ..., j_{u_i} \in \{1, 2, ..., n\}$. Подмножества $C_1, C_2, ..., C_r$ могут пересекаться и $\bigcup_{i=1}^{r} C_i = \{1, ..., n\}$. Здесь n – число входов в схеме. Пусть также N_1 – максимальное число псевдослучайных наборов, которые используются для формирования начального отрезка теста T; N_2 – максимальное число входных наборов, генерируемых в процессе одной итерации на основе генетического алгоритма; N_3 – максимальное число итераций.

АЛГОРИТМ

1. Последовательно генерируются N_1 псевдослучайных входных наборов. Для каждого входного набора t_i находится множество ранее необнаруженных неисправностей, которые проверяются данным набором, и определяется их число k_i (данная задача решается с помощью моделирования неисправностей). Если $k_i > 0$, то набор t_i включается в тест T. Если в процессе генерации псевдослучайных наборов окажется, что список еще необнаруженных неисправностей становится пустым, то выполнение алгоритма завершается.

$I := 1$.

2. Наборы из теста T упорядочиваются в порядке убывания числа обнаруживаемых ими неисправностей. Формируется множество наборов T_1. При этом в T_1 включаются наборы с лучшим качеством (качество набора определяется числом проверяемых им неисправностей; чем больше это число, тем выше качество набора). При этом $|T_1| = [|T| / 3]$.

$K := 1$.

3. Генерируется случайное число $p \in [0, 1]$. Если $p \le 0{,}5$, 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.

Если наборы выбираются из T_1, то генерируются два случайных числа $p_1, p_2 \in [0, 1]$, $p_1 \ne p_2$. Если $(i - 1)(1 / |T_1|) \le p_k < (1 / |T_1|)i$, то i-й набор из T_1 выбирается в качестве t_k. Здесь $k \in \{0, 1\}$. Если p_2 приводит к выбору того же входного набора, что и p_1, то генерируется другое псевдослучайное число из отрезка [0, 1], чтобы исключить такой случай.

Аналогичным образом выбираются наборы и из T (если $p > 0.5$).

4. Случайным образом упорядочиваются подмножества C_1, C_2, ..., C_r из C. Это можно сделать следующим образом. Генерируется случайное число $p \in [0, 1]$. Если $(i-1)(1/r) \le p < (1/r)i$, то C_i помещается в конец формируемой последовательности и C_i удаляется из C (вначале последовательность не содержит подмножеств из C). Далее указанная операция повторяется для нового C с учетом того, что $|C| = r - 1$. Процесс продолжается до тех пор, пока $C \neq \varnothing$.

В результате получим последовательность $C^u = (C_{i_1}, C_{i_2}, ..., C_{i_r})$; $i_1, i_2, ..., i_r \in \{1, 2, ..., r\}$.

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 $p \le 0.5$, 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 $p_1, p_2, ..., p_n \in [0,1]$. Если $p_j \le 1 / n$, 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 $T := T \cup T_I$ 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.

В рассматриваемых программных средствах принята единая структура данных в виде совокупности массивов и файлов, содержащих описание схемы и результаты построения и анализа на полноту теста для нее. Наличие инвариантного ядра в данных средствах, включающего универсальную структуру данных и единый интерфейс, позволяет осуществлять их развитие путем модификации имеющихся решающих программных модулей и подключения новых.

На базе данных программных средств выполняются лабораторные работы, позволяющие получить практические навыки автоматизированного построения тестов для современных БИС/СБИС, а также курсовые и дипломные работы, связанные с изучением и исследованием методов анализа и тестирования БИС/СБИС.

ЛИТЕРАТУРА

1. *Люлькин, А. Е.* Состав и организация учебно-исследовательской системы автоматизированного построения тестов для дискретных устройств на ПЭВМ / А. Е. Люлькин // Управляющие системы и машины. – 1996. – № 1–2. – С. 48–55.
2. *Люлькин, А. Е.* Моделирование неисправностей в функционально-переключательных КМОП-структурах / А. Е. Люлькин // Электронное моделирование. – 1998. – № 5. – С. 49–59.