УДК 004.312+004.023

А.Ю. Городилов

Пермский государственный национальный исследовательский университет, Пермь, Россия

ГЕНЕТИЧЕСКИЙ АЛГОРИТМ ДИАГНОСТИРОВАНИЯ ЦИФРОВЫХ УСТРОЙСТВ

Описан генетический алгоритм, с помощью которого решается задача построения диагностической последовательности для программируемых логических интегральных схем на основе элементов с избыточным базисом. С помощью диагностической последовательности возможно обнаружить отказ, выполнить реконфигурацию схемы и восстановить работоспособность устройства, тем самым повышая его надежность. Рассмотрены различные виды отказов элементов с избыточным базисом и способы их диагностирования, приведена математическая постановка задачи диагностирования заданной конфигурации интегральной схемы. Для повышения коэффициента готовности цифрового устройства необходимо минимизировать длину диагностической последовательности. Для решения этой оптимизационной задачи применены генетические алгоритмы. Предложена оригинальная схема применения алгоритма, включающая дополнительный внешний цикл, которая позволила решить проблему многокритериальности рассматриваемой задачи и увеличить точность получаемого решения за счет использования стандартных генетических операторов. Приведены результаты экспериментов для модельных задач.

Ключевые слова: программируемые логические интегральные схемы, диагностика, надежность, генетический алгоритм, оптимизация.

A.Yu. Gorodilov

Perm State National Research University, Perm, Russian Federation

GENETIC ALGORITHM OF DIGITAL DEVICES DIAGNOSIS

The paper deals with the description of genetic algorithm aimed at producing diagnosing sequence for excess components based programmable logic. Diagnosing sequence makes it possible to detect circuit re-configuration failure, to renovate the device full capability and to enhance the device reliability. Various types of components failures are studied and some diagnosing ways are proposed, as well as the mathematical way for a particular integrated circuit configuration testing. In order to improve the digital device availability it is necessary to minimize the diagnosing sequence duration. It is possible to optimize the diagnosis process due to genetic algorithm use. There has been developed an original scheme of algorithm application including supplementary outer loop that leads to solving the problem of multicriteriality and to the resulting accuracy increase due to standard genetic operators application.The results of the experiments for model tasks are described.

Keywords: programmable logic circuitry, diagnosis, reliability, genetic algorithm, optimization.

При проектировании и эксплуатации цифровых устройств часто возникает задача проверки их работоспособности, то есть способности выполнять заданные функции. Особенно актуальной такая задача является для устройств, построенных на базе программируемых логических интегральных схем (ПЛИС). Особенностью ПЛИС является возможность их перепрограммирования, то есть изменения логики их работы и схемы размещения логических блоков на кристалле. В случае обнаружения неисправности отказавший элемент может быть выведен из схемы и заменен другим, ранее незадействованным резервным элементом, тем самым работоспособное состояние устройства будет восстановлено. Целью данной работы является разработка алгоритма построения диагностической последовательности ПЛИС, то есть такой последовательности входных сигналов, что на основе анализа выходных сигналов диагностируемого устройства можно будет обнаружить отказ, а также определить место, где он произошел, и его тип. Для достижения поставленной цели предлагается использовать метод генетических алгоритмов.

Задача диагностирования

Одной из основных составляющих в ПЛИС типа FPGA (Field Programmable Gate Array – программируемая пользователем вентильная матрица) являются конфигурируемые логические блоки. Мы будем рассматривать отказы логических элементов в них. Для повышения отказоустойчивости в [1] предлагается использовать элементы с избыточным базисом в качестве основы FPGA. В случае возникновения однократных константных отказов на входе такой элемент сохраняет частичную функциональность и может быть задействован в схеме после реконфигурации. Таким образом, в каждый момент времени элемент с избыточным базисом может находиться в одном из следующих состояний:

0. Полностью работоспособное состояние, отказов нет. Элемент реализует избыточный базис, состоящий из одной функции $f_0 = \overline{x_1} \cdot \overline{x_2} \vee \overline{x_3} \cdot \overline{x_4}$.

1. Отказ типа x_1=0. Элемент реализует остаточный базис $f_1 = \overline{x_2} \vee \overline{x_3} \cdot \overline{x_4}$.

2. Отказ типа x_2=0. Элемент реализует остаточный базис $f_2 = \overline{x_1} \vee \overline{x_3} \cdot \overline{x_4}$.

3. Отказ типа x_3=0. Элемент реализует остаточный базис $f_3 = \overline{x_1} \cdot \overline{x_2} \vee \overline{x_4}$.

4. Отказ типа x_4=0. Элемент реализует остаточный базис $f_4 = \overline{x_1} \cdot \overline{x_2} \vee \overline{x_3}$.

5. Отказ типа x_1=1. Элемент реализует остаточный базис $f_5 = \overline{x_3} \cdot \overline{x_4}$.

6. Отказ типа x_2=1. Элемент реализует остаточный базис $f_5 = \overline{x_3} \cdot \overline{x_4}$.

7. Отказ типа x_3=1. Элемент реализует остаточный базис $f_6 = \overline{x_1} \cdot \overline{x_2}$.

8. Отказ типа x_4=1. Элемент реализует остаточный базис $f_6 = \overline{x_1} \cdot \overline{x_2}$.

9. Иной отказ либо комбинация нескольких отказов. Элемент, находящийся в таком состоянии, не может быть использован в дальнейшем.

В [2] предлагается алгоритм диагностирования ПЛИС, суть которого состоит в последовательном тестировании каждой ячейки и определении ее состояния. Однако недостатком такого подхода можно считать необходимость введения специального средства диагностического разрыва (мультиплексора) в каждую ячейку, а также усложнение структуры программируемых межсоединений, что ведет к увеличению аппаратных ресурсов, не задействованных непосредственно в реализации схемы цифрового устройства.

Предлагаемый в данной статье метод заключается в построении диагностической последовательности для текущей конфигурации ПЛИС. Под диагностической последовательностью здесь понимается серия наборов входных сигналов, подаваемых на вход блоков ввода/вывода ПЛИС. При различных вариантах отказов на выходе блоков будут получаться различные наборы выходных сигналов. Анализируя эти сигналы, можно определить, произошел ли отказ, и если да, то какой именно. Для загрузки последовательности входных наборов и считывания выходных наборов можно воспользоваться стандартным интерфейсом JTAG [3], который поддерживается большинством производителей ПЛИС.

Пусть в текущей конфигурации задействовано N логических элементов. Будем считать вероятность одновременного отказа двух и более элементов пренебрежимо малой. Тогда количество различных неисправных модификаций схемы равно $n = 9N$ (один из 9 видов отказов, описанных выше, происходит в одном из N элементов). Обозначим через A_0 исправное цифровое устройство, а через $A = \{A_1, A_2, ..., A_n\}$ – множество неисправных модификаций, порождаемых рассматриваемым классом неисправностей. Пусть $x = (x(1),x(2),\ldots,x(t))$ – последовательность наборов входных сигналов. Если подать эту последовательность на вход устройства A_i, на выходе мы получим последовательность выходных сигналов $y_i = A_i(x) = (y_i(1),y_i(2),\ldots,y_i(t))$. Если y_0 отличается от всех y_i, $i = 1..n$, то последовательность x называется тестовой, и мы будем говорить, что диагностический тест x обнаруживает неисправность. Если для некоторого j последовательность y_j отличается от всех y_i, $i = 0..n$, $i \neq j$, то диагностический тест x точно идентифицирует неисправность j, то есть определяет место возникновения и тип отказа. Для некоторых пар неисправностей может оказаться, что для любой входной последовательности соответствующие выходные последовательности будут совпадать, тогда такие неисправности отличить в текущей конфигурации невозможно, и для их однозначной идентификации потребуется реконфигурация схемы. То есть даже полный тест (перебор всех возможных входных последовательностей) может не позволить однозначно определить место и тип отказа. Также нужно отметить, что количество наборов входных сигналов в диагностической последовательности (длина последовательности t) определяет время диагностирования. Чем короче будет диагностическая последовательность, тем быстрее будет выполняться диагностирование и, следовательно, быстрее можно будет восстановить работоспособное состояние устройства.

Таким образом, для диагностирования ПЛИС необходимо решить задачу построения кратчайшего диагностического теста, позволяющего обнаружить неисправность и наиболее точно идентифицировать место ее возникновения (указать конкретный логический элемент) и тип отказа. Для решения такой оптимизационной задачи нами предлагается использовать метод генетических алгоритмов.

Генетический алгоритм

Для генерации диагностических тестов генетические алгоритмы (ГА) применяются, начиная с середины 90-х гг. В первых работах [4], где для решения задачи построения тестов используются ГА, особи представляли собой простые двоичные векторы (наборы входных сигналов), а не тестовые последовательности. Функция приспособленности для такой особи рассчитывалась, как количество однозначно идентифицируемых неисправностей, которые мог выявить данный тест. Операции скрещивания и мутации ничем не отличались от канонического ГА. Позднее появились работы [5], где в качестве особи ГА рассматривалась двоичная матрица, соответствующая входной диагностической последовательности. Такое кодирование на данный момент считается стандартным на логическом уровне представления цифровых устройств.

Дальнейшим развитием ГА стало появление их параллельных версий. Параллельные модели ГА в области построения диагностических тестов не отличаются принципиально от параллельных реализаций ГА, применяемых для решения других задач – параллельное вычисление фитнес-функции и островная модель. Обычно для тестирования этих алгоритмов используются слабопараллельные (2–4-ядерные процессоры) и сильнопараллельные (8 и более ядер) вычислительные системы. Но в последнее время появились данные об использовании технологии CUDA многоядерных графических ускорителей. Хотя в последнем случае возможна лишь мелкоячеестая структура распараллеливания, но коэффициент ускорения вычислений может быть весьма значительным.

Кроме непосредственного получения тестовых последовательностей ГА может быть использован для сопутствующих задач, например, для удаления избыточной информации из диагностических словарей [6].

Особенностью исследуемой задачи является, во-первых, рассмотрение комбинационных схем, в то время как большинство исследователей решают задачу построения тестов для последовательностных схем [7]. В последнем случае для построения дерева контроля необходимо для каждого узла искать наилучшую тестовую последовательность. Для комбинационных же схем каждому узлу дерева будет соответствовать один входной набор, то есть последовательность наборов

будет задавать все дерево целиком. Таким образом, предлагаемый нами алгоритм находит все множество наборов, необходимых для диагностирования устройства, в отличие от известных генетических алгоритмов, осуществляющих поиск только для одного узла дерева контроля. Во-вторых, особенность задачи состоит в используемой модели отказов, построенной для элементов с избыточным базисом, в отличие от известных работ, рассматривающих стандартные вентили 2И-НЕ, 2ИЛИ-НЕ, либо вообще не учитывающих специфику элементной базы.

Для разработки генетического алгоритма нам нужно решить, как будет представляться особь, как вычислять значение фитнес-функции, определить операторы селекции, скрещивания, мутации. Ключевым здесь является выбор способа представления.

Как уже было отмечено, в рассматриваемой задаче решением является двоичная матрица, соответствующая входной диагностической последовательности, следовательно, особью в генетическом алгоритме должна быть двоичная матрица. Количество столбцов в матрице соответствует количеству входных сигналов схемы, то есть длине входных наборов. Количество строк в матрице соответствует количеству наборов, то есть длине входной последовательности. Длина всех входных наборов одинакова, а их количество в различных последовательностях может быть различным.

В результате двоичные матрицы могут иметь разное количество строк, что приведет к невозможности использования стандартного оператора скрещивания. Вместе с тем согласно исследованию [8] использование стандартных генетических операторов в целом повышает качество получаемого решения. По этой причине изменим структуру генетического алгоритма, добавив внешний цикл, в котором будем устанавливать различные длины последовательностей, постепенно увеличивая их. Такой подход позволит также устранить проблему многокритериальностью рассматриваемой задачи и в рамках каждой итерации оптимизировать только количество идентифицируемых отказов. Как только увеличение длины последовательности перестанет приводить к увеличению числа идентифицируемых отказов, это с высокой вероятностью будет означать, что лучшие показатели являются недостижимыми, и выполнение алгоритма на этом можно будет остановить. Общая схема полученного алгоритма приведена на рисунке.

Как было изложено выше, такая схема позволяет тривиально вычислять приспособленность особей. В качестве значения фитнес-функции $f(x)$ берется количество точно идентифицированных отказов, диагностированных последовательностью x.

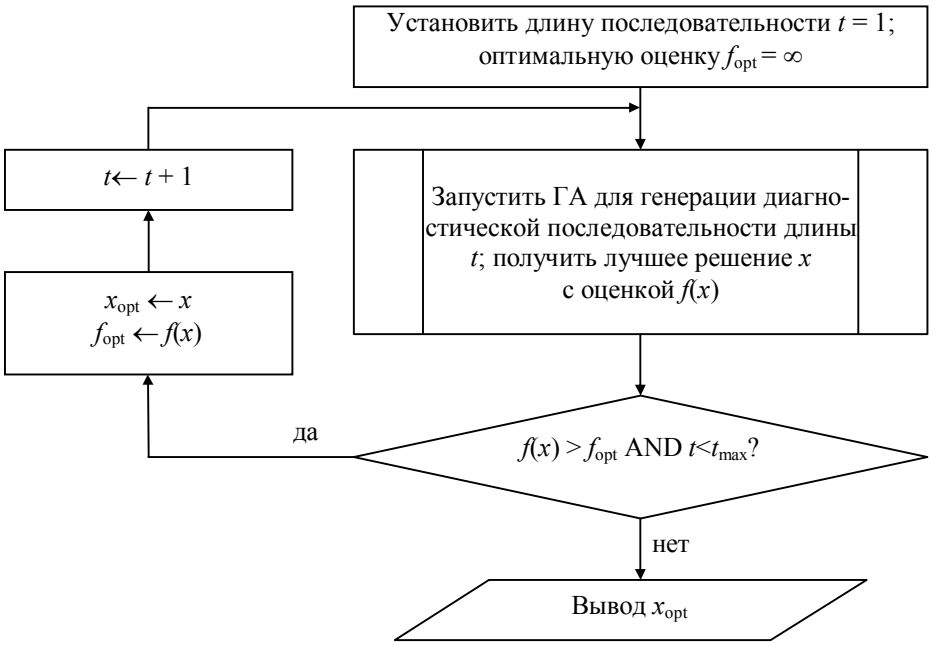

Рис. Схема применения генетического алгоритма

Уточним остальные используемые операторы. Будем использовать два вида одноточечного оператора скрещивания – вертикальный и горизонтальный. В первом случае множество столбцов обеих матриц делится точкой разреза на две части. Первый потомок получается объединением первой части столбцов первой матрицы и второй части столбцов второй матрицы, второй потомок – объединением первой части второй матрицы и второй части первой матрицы. В случае горизонтального оператора скрещивания аналогичные действия выполняются для множества строк. В качестве оператора мутации будем использовать стандартный одноточечный оператор: случайно выбранная строка матрицы заменяется сгенерированной случайным образом. Оператор селекции базируется на идеях выбора родителей по принципу «рулетки» и включения потомков в популяцию на основе принципов «элитизма» и постоянства численности популяции. Условием окончания является достижение определенного номера популяции.

Для рассматриваемой задачи построения кратчайшего диагностического теста был разработан генетический алгоритм, особенностью которого является использование внешнего цикла для перебора различных вариантов длин тестов и использование стандартных генетических операторов. Применение использованных подходов на модельных задачах оптимизации (поиска кратчайшего цикла в графе, поиска компактных групп объектов и т.п.) показывает, что генетический алгоритм, использующий стандартные операторы, позволяет найти решение, более близкое к оптимальному, в 70–80 % случаев (в задаче поиска кратчайшего цикла).

Библиографический список

1. Тюрин С.Ф., Громов О.А. Базисный элемент программируемых логических интегральных схем // Вестник Ижевск. гос. техн. ун-та. – 2010. – № 3. – С. 122–125.

2. Тюрин С.Ф., Громов О.А. Разработка контрольных и диагностических тестов для КМОП элементов с избыточным базисом // Приволжский научный вестник. – 2013. – № 1(17). – С. 13–21.

3. Каршенбойм И. JTAG-тестирование // Современная электроника. – 2007. – № 2. – С. 58–66.

4. Sequential Circuit Test Generation in a Genetic Algorithm Framework / E.M. Rudnick, J.H. Patel, G.S. Greenstein, T.M. Niermann // Proc. Design Automation Conf. – 1994. – P. 698–704.

5. Prinetto P., Rebaudengo M., Sonza Reorda M. An automatic test pattern generator for large sequential circuits based on genetic algorithms // Proc. Int. Test Conf. – 1994. – P.240–249.

6. Миронов С.В., Сперанский Д.В. Генетические алгоритмы для сокращения диагностической информации // Автоматика и телемеханика. – 2008. – № 7. – С.146–156.

7. Иванов Д.Е. Генетические алгоритмы построения входных идентифицирующих последовательностей цифровых устройств. – Донецк: ТОВ «Цифровая типографія», 2012. – 240 с.

8. Данилова Е.Ю., Городилов А.Ю. Сравнение генетических алгоритмов на примере задачи коммивояжера // Вестник Перм. гос. техн. ун-та. Сер. Математика. Механика. Информатика. – 2009. – Вып. 3(29). – С. 49–53.

Сведения об авторе

Городилов Алексей Юрьевич (Пермь, Россия) – старший преподаватель кафедры математического обеспечения вычислительных систем Пермского государственного научного исследовательского университета (614990, г. Пермь, ул. Букирева, 15, e-mail: gora830@yandex.ru).

About the author

Gorodilov Aleksey Iurievich (Perm, Russian Federation) is Senior Lecturer at the Department of Mathematical Support for Computing Systems, Perm State National Research University (614990, 15, Bukirev St., Perm, e-mail: gora830@yandex.ru).

Получено 06.09.2013