ДВУХУРОВНЕВЫЙ ГЕНЕТИЧЕСКИЙ АЛГОРИТМ РЕКОНФИГУРАЦИИ ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ ИНТЕГРАЛЬНЫХ СХЕМ

Алексей Городилов

***Аннотация**: В статье рассматривается задача реконфигурации программируемой логической интегральной схемы после возникновения отказов логических элементов. Задача является актуальной для таких областей, как космонавтика и управление ответственными промышленными объектами, в которых используются высоконадежные отказоустойчивые системы. В статье рассматриваются отказы логических элементов в конфигурируемых логических блоках. Для оптимальной реконфигурации необходимо разместить логические модули на программируемой логической интегральной схеме, не используя отказавшие логические элементы, при условии минимизации занимаемой площади, длины связей и величины задержек. Первым этапом в решении поставленной задачи является выбор компактно расположенных логических элементов, которые будут задействованы в реализации схемы. Компактность здесь понимается как минимальность количества информации, необходимой для описания используемого множества логических элементов. Поставленная задача не может быть решена точно и эффективно, но может быть решена приближенно с помощью различных эвристических методов, в том числе генетических алгоритмов. В статье описывается несколько известных подходов к решению задачи и предлагается оригинальный двухуровневый способ кодирования для построения генетического алгоритма, учитывающего особенности задачи и использующего стандартные генетические операторы. Предложенный алгоритм реализован на практике в виде компьютерной программы, позволяющей сравнивать его работу с некоторыми другими алгоритмами. Выполнено сравнение различных эвристических алгоритмов с точки зрения точности находимого решения, оценивается скорость работы предложенного алгоритма. Можно сказать, что предложенный генетический алгоритм в среднем работает дольше других эвристических алгоритмов, но находит более точное решение, что ведет к уменьшению времени реконфигурации и, тем самым, к увеличению коэффициента готовности программируемой логической интегральной схемы.*

***Ключевые слова**: программируемые логические интегральные схемы, реконфигурация, надежность, генетический алгоритм.*

***ACM Classification Keywords**: I.2 Artificial Intelligence: I.2.8 Problem Solving, Control Methods, and Search - Heuristic methods, B.8 Performance and Reliability: B.8.1 Reliability, Testing, and Fault-Tolerance*

Введение

Современные цифровые устройства содержат множество транзисторов и линий соединений, отказ которых может привести к нарушению работы всего устройства, поэтому для обеспечения бесперебойной работы оборудования в высоконадежных системах (например, в области космонавтики и управления ответственными промышленными объектами) требуются соответствующие методы проектирования, диагностирования и восстановления. Для решения задачи восстановления могут быть использованы программируемые логические интегральные схемы (ПЛИС), которые предоставляют широкие

возможности логической реконфигурации. ПЛИС являются универсальным базисом для проектирования цифровых устройств любого уровня сложности. По статистическим данным ведущих фирм-производителей объем производства ПЛИС постоянно увеличивается [Уваров, 2007].

При реконфигурации основной задачей является оптимальное выполнение стадий размещения и соединения с учетом частичной или полной неработоспособности отказавших элементов. Оптимальность здесь понимается в смысле минимизации занятой площади, длины соединений и величины задержек. Для решения сформулированной задачи оптимизации нами предлагается использовать генетический алгоритм.

Задача реконфигурации

Каждый производитель ПЛИС имеет свою собственную архитектуру, но в целом все они являются вариациями архитектуры, показанной на рис. 1 [Zeidman, 2002].

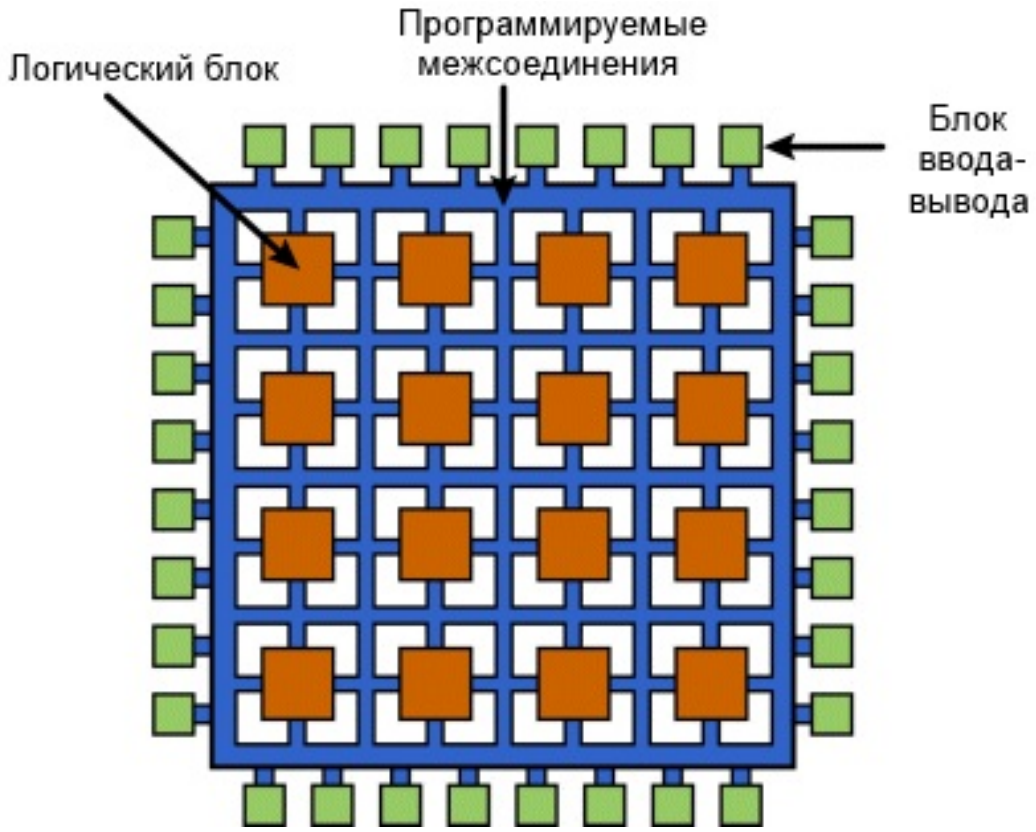

Рис. 1. Общая схема ПЛИС типа FPGA

Основными составляющими в ПЛИС типа FPGA являются ячейки конфигурационной памяти, конфигурируемые логические блоки (КЛБ), блоки ввода-вывода и программируемые межсоединения.

В данной статье рассматриваются отказы логических элементов (ЛЭ) в КЛБ. Некоторые виды отказов могут приводить к полному выходу из строя элемента и невозможности его дальнейшего использования. Такие элементы должны быть исключены и в результате реконфигурации их функции будут реализованы другими элементами. Однако, некоторые отказы приводят к тому, что элемент сохраняет способность

выполнять какую-либо полезную работу. Например, к таким видам отказов относятся однократные константные отказы для некоторых видов логических элементов: элементов с избыточным базисом [Tyurin, 1999, Tyurin, 2011], 8-1 мультиплексоров и таблиц перекодировки (LUT) [Tyurin, 2013]. В случае такого отказа КЛБ может быть задействован в схеме, но с учетом его ограниченной функциональности.

Согласно [Цыбин, 2006], большая доля отказов приходится на ячейки памяти. Однако, такие отказы тоже можно рассматривать как однократные константные отказы на входе логических элементов: если ячейка памяти вышла из строя, она генерирует на выходе какое-либо постоянное значение (0 или 1). Отказы в соединительных линиях также могут быть описаны константными отказами в логических элементах (обрыв соединения означает константу 0, замыкание на питание – константу 1).

По результатам диагностирования должна быть получена информация о состоянии каждого логического элемента. Поскольку элементы в ПЛИС образуют строго упорядоченную матричную структуру (см. рис. 1), эти данные можно представить в виде таблицы, значения которой обозначают состояние соответствующих логических элементов (отказов нет, полностью неработоспособен, либо частично работоспособен, с указанием его ограниченных возможностей). Такую таблицу будем называть таблицей состояний. Задача реконфигурации состоит в поиске новой схемы размещения и соединения ЛЭ, для чего, в первую очередь, нужно выбрать из доступного множества элементов те, которые будут использованы в схеме.

Нужно отметить, что традиционная архитектура обычно содержит систему локальных соединений нижнего уровня и глобальные линии связи, которые в совокупности обеспечивают возможность установления связи между любой парой логических элементов. Однако для снижения затрат конфигурационной памяти в FPGA может быть реализована трехуровневая структура, в которой прямая связь между некоторыми удаленными элементами может быть недоступна [Цыбин, 2006]. Кроме того, соединение между логическими элементами имеет характеристики (например, количество межсоединительных переключателей в цепи, влияющее на величину задержки сигнала), значение которых нужно минимизировать. В общем случае можно считать, что более предпочтительным является такой выбор ЛЭ, при котором они будут располагаться наиболее компактно.

Множество используемых ЛЭ в системах конфигурации ПЛИС описывается набором прямоугольников (координатами их левого верхнего угла и размерами). Прямоугольники не должны пересекаться и не должны содержать неработоспособных элементов. Чем меньше количество таких прямоугольников в описании, тем короче будет конфигурационный файл, и тем быстрее будет выполнена реконфигурация. Кроме того, элементы внутри одного прямоугольника располагаются максимально компактно, поэтому минимизация количества прямоугольников ведет и к оптимизации расположения элементов, то есть желаемой минимизации характеристик соединений.

Рассмотрим упрощенный случай, когда каждый элемент может находиться только в одном из двух состояний: полностью работоспособном (будет обозначаться «1» в таблице состояний) или полностью неработоспособном (будет обозначаться «0»). Назовем прямоугольный фрагмент таблицы состояний, полностью состоящий из «1» покрывающим прямоугольником. Пусть N – количество ЛЭ, необходимых для реализации схемы устройства (или, по крайней мере, той части устройства, которая обеспечивает выполнение основных и наиболее важных функций). Тогда первый этап в решении задачи реконфигурации заключается в поиске для заданной таблицы состояний минимального числа покрывающих прямоугольников, содержащих в сумме не менее N элементов. Поскольку количество логических элементов в FPGA может быть очень велико, пространство поиска получается большим.

Подходы к решению

Тривиальный подход к решению задачи заключается в переборе различных наборов покрывающих прямоугольников. Однако, общее их количество в худшем случае пропорционально числу элементов в матрице. Таким образом, количество наборов, то есть перебираемых вариантов, будет расти экспоненциально с ростом числа элементов. Следовательно, тривиальный алгоритм является неэффективным и не может использоваться на реальных данных.

Поскольку точных эффективных алгоритмов решения поставленной задачи не известно, на практике применяют различные эвристические алгоритмы. Одним из них является следующий «жадный» алгоритм:

1. Установить общее количество искомых элементов: $Need \leftarrow N$.
2. Найти покрывающий прямоугольник R, содержащий наибольшее количество «1» N_R.
3. Установить все элементы таблицы в прямоугольнике R в значение «0».
4. Уменьшить количество искомых элементов: $Need \leftarrow Need - N_R$.
5. Если найденных элементов не достаточно ($Need > 0$), перейти к шагу 2.

Данный алгоритм работает быстро, но не всегда находит точное решение, то есть не всегда найденное количество покрывающих прямоугольников оказывается минимальным.

Для решения указанной проблемы в данной статье предлагается использовать генетические алгоритмы, также являющиеся разновидностью эвристических алгоритмов. Их использование в рассматриваемой задаче представляется оправданным, поскольку генетические алгоритмы хорошо зарекомендовали себя для задач, где пространство поиска велико и не монотонно [Гладков, 2006].

Известны примеры применения генетических алгоритмов в задачах проектирования и диагностирования ПЛИС. Так, в [del Solar, 2006] ГА используются для задач размещения и связи. Однако, рассматриваемая в данной статье задача реконфигурации существенно отличается наличием неработоспособных элементов в матрице, что существенно увеличивает сложность задачи размещения. В [Yakimets, 2011] рассматриваются альтернативные подходы к разработке систем ПЛИС на основе ГА. Предлагается использовать несколько генотипов на одном (или нескольких) кристаллах ПЛИС и разрабатывать частично корректные детерминированные автоматы. Их использование уменьшает временные затраты на эволюцию, однако усложняет методику генерации проекта.

В данной статье предлагается доработка стандартного генетического алгоритма, которая позволит учесть дополнительные ограничения и повысить качество работы алгоритма. Отличительной особенностью предлагаемой доработки является новый способ кодирования хромосом.

Генетический алгоритм

Для разработки генетического алгоритма необходимо определить структуру хромосом и функцию приспособленности. Поскольку решением рассматриваемой задачи является набор покрывающих прямоугольников, тривиальным подходом является структура хромосом в виде последовательности дескрипторов прямоугольников. Однако, такой подход порождает множество проблем с генетическими операторами, поскольку хромосомы будут иметь различную длину в зависимости от количества прямоугольников в кодируемом ими решении. По этой причине мы будем использовать другой подход.

Для каждого элемента таблицы определим покрывающий прямоугольник максимальной площади, для которого данный элемент является левым верхним углом. Назовем такие покрывающие прямоугольники

локально-максимальными. Справедливо утверждение, что существует такая последовательность локально-максимальных прямоугольников, что их последовательный выбор приведет к точному решению задачи. Следовательно, решение может быть найдено в виде последовательности локально-максимальных прямоугольников, представленных своими дескрипторами. Таким образом, хромосома будет представлять собой перестановку на множестве дескрипторов локально-максимальных прямоугольников. При таком подходе длина каждой хромосомы L будет фиксирована и равна количеству локально-максимальных прямоугольников, что соответствует числу «1» в таблице состояний.

Однако, равная длина всех хромосом не решает всех проблем с генетическими операторами. Если считать отдельными генами хромосомы непосредственно элементы перестановки P, то есть i-ым геном особи считать число p_i, то гены получаются взаимозависимыми: если какой-то ген равен j, то никакой другой ген этой особи не может принимать значение j. Указанная интерпретация интуитивно понятна и не требует дополнительных вычислительных затрат, однако зависимость генов приводит к тому, что стандартные операторы оказываются неприменимы, так как все они работают с представлением набора генов в виде строки независимых бит.

Альтернативным подходом, предлагаемым в данной статье, является использование промежуточного представления особей в виде некоторого объекта, легко трансформируемого в перестановку. Такой объект должен задаваться при помощи битовой или числовой строки или матрицы, состоящей из независимых элементов.

Перестановке соответствует полное паросочетание в полном двудольном графе [Aho, 1987]. Пусть C – матрица размера LxL. Элемент c_{ij} соответствует весу ребра между i-той вершиной первой доли и j-той вершиной второй доли в графе G. Зная такую матрицу, можно найти полное паросочетание минимального веса в графе G. Такая задача решается достаточно быстро за полиномиальное время [Асанов, 2001]. Упорядочим ребра паросочетания по номеру вершины первой доли. Тогда номера вершин второй доли образуют перестановку на множестве чисел (1..L).

Таким образом, хромосомой в генетическом алгоритме является матрица C, а процесс кодирования является двухуровневым.

1. Строится двудольный граф с весами ребер, равными элементам матрицы C, в графе ищется полное паросочетание минимального веса.
2. Получившееся паросочетание определяет перестановку, которая задает порядок выбора локально-максимальных прямоугольников.

Приспособленность особи, как и окончательное решение задачи, определяются на основе алгоритма, аналогичного рассмотренному выше «жадному»:

1. Установить общее количество искомых элементов: $Need \leftarrow N$, количество используемых прямоугольников $N_{rect} \leftarrow 0$.
2. Взять очередной локально-максимальный прямоугольник R из перестановки, количество «1» в нем N_R.
3. Установить все элементы таблицы в прямоугольнике R в значение «0».
4. Уменьшить количество искомых элементов: $Need \leftarrow Need - N_R$, увеличить количество используемых прямоугольников $N_{rect} \leftarrow N_{rect} + 1$.
5. Если найденных элементов не достаточно ($Need > 0$), перейти к шагу 2.

Приспособленность особи будет равна количеству использованных прямоугольников N_{rect}.

Независимость элементов матрицы позволяет использовать стандартные генетические операции мутации и скрещивания. При селекции особей используются две основные классические идеи: «принцип рулетки» и «принцип элитизма» [Mitchell, 1999]. Размер популяции остается постоянным за счет удаления наименее приспособленных особей на каждом шаге. Условием окончания работы предлагаемого генетического алгоритма является достижения определенного поколения с номером *M*. Это позволяет заранее оценить время работы алгоритма или, наоборот, задать параметр *M* так, чтобы алгоритм завершил свою работу за указанное фиксированное время.

Предложенный алгоритм может быть легко адаптирован для общего случая, когда таблица состояний содержит и другие значения, соответствующие ЛЭ в частично работоспособном состоянии. В этом случае изменяется только этап определения локально-максимальных прямоугольников: элементы с несовместимым остаточным базисом [Tyurin, 2011] не могут попасть в один блок.

Реализация и пример работы

Для экспериментального сравнения различных алгоритмов была создана программная реализация приведенных выше «жадного» и генетического алгоритмов (рис. 2). Для генетического алгоритма предусмотрена возможность задания параметров, которые существенно влияют на качество находимого решения. К таким параметрам относятся

- численность популяции;
- вероятность мутации;
- процент лучших особей, переходящих в новое поколение без изменений;
- процент особей, принимающих участие в скрещивании в каждом поколении;
- количество поколений.

Значения параметров определяются экспериментально, поскольку общих теоретических рекомендаций по их подбору не существует. Очевидно, значение параметров численности популяции и количества поколений должно зависеть от объема исходных данных.

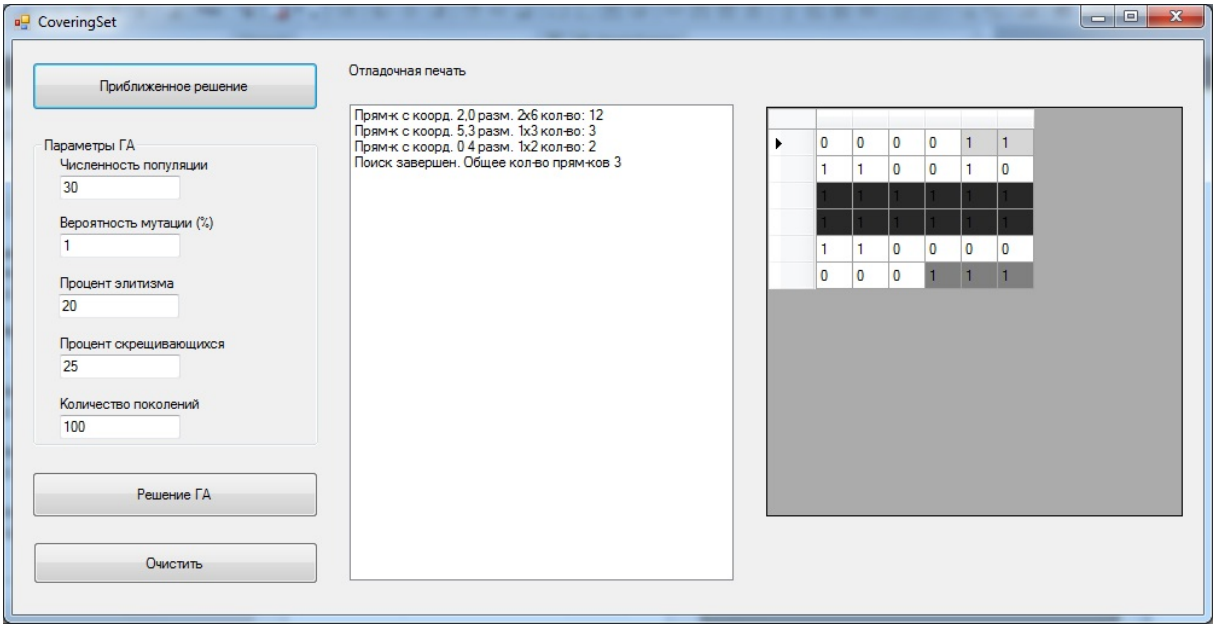

Рис. 2. Программная реализация алгоритмов. Приближенное решение

На рис. 2 в правом окне показано решение, которое находит «жадный» алгоритм (значения таблицы состояний можно видеть на рисунке, N=16). Данное решение, использующее 3 прямоугольника, не является оптимальным, поскольку 16 единиц можно покрыть двумя прямоугольниками, если не выделять целиком весь максимальный прямоугольник. Оптимальное решение можно увидеть на рис. 3.

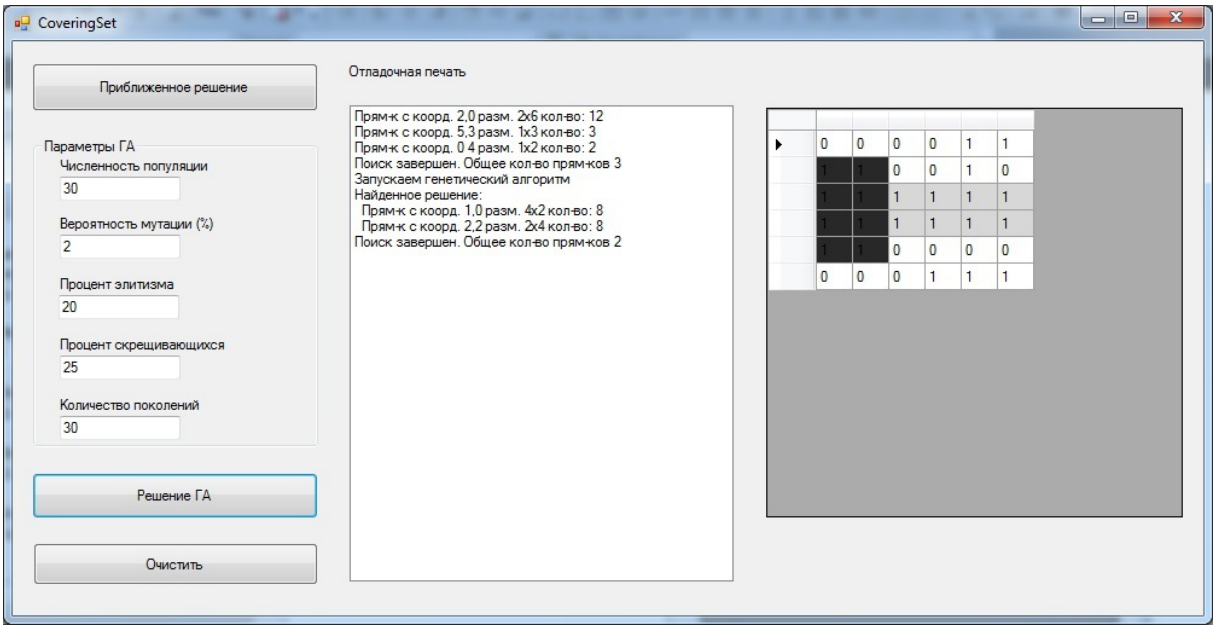

Рис. 3. Точное решение, найденное генетическим алгоритмом

Для генетического алгоритма был выполнен подбор оптимальных параметров, при которых алгоритм устойчиво находил оптимальное решение. Значения этих параметров, а также результат работы алгоритма можно увидеть на рис. 3.

Оценка эффективности

Время работы генетического алгоритма существенно зависит от значений параметров, поэтому сравнение его эффективности с другими алгоритмами затруднительно. Если подобрать параметры, при которых генетический алгоритм стабильно находит точное решение, то время его работы получается в несколько раз больше, чем время работы «жадного» алгоритма.

Выполним оценку количества операций, выполняемых генетическим алгоритмом, по сравнению с алгоритмом, осуществляющим полный перебор. Количество операций в обоих алгоритмах определяется количеством анализируемых решений. Рассмотрим случай таблицы размером 6x6, содержащей q=22 единиц (рис. 3). Для оценки количества операций в генетическом алгоритме будем использовать найденные значения параметров: численность популяции N_{pop}=30, количество поколений N_{gen}=30. Для полного перебора количество вариантов определяется формулой

$$op_{\text{перебора}} = 2^q = 2^{22} \sim 10^6$$

Для генетического алгоритма число анализируемых решений определяется количеством рассмотренных особей и может быть вычислено следующим образом

$$op_{ГА} = N_{pop}*N_{gen} = 30*30 \sim 10^3$$

Таким образом, генетический алгоритм рассматривает существенно меньше решений, за счет чего работает принципиально быстрее, по-прежнему находя в большинстве случаев точное оптимальное решение.

Для оценки эффективности предложенного подхода кодирования особей, были рассмотрены несколько модельных задач на графах [Данилова, 2009]. Для всех задач двухуровневый генетический алгоритм чаще находил решение, более близкое к оптимальному. Например, для задачи коммивояжера ГА с модифицированным кодированием примерно в 1.5 раза чаще находит решение с наименьшей длиной цикла, что соответствует оптимальному решению. То есть, ГА с двухуровневым кодированием с большей вероятностью дает достаточно хороший результат, следовательно, позволяет найти некоторое приемлемое решение быстрее.

В задачах, где можно было найти точное решение, оценивался процент запусков ГА, в которых это точное решение было найдено. Результаты для задачи разбиения графа на максимально независимые подграфы можно увидеть в табл. 1. Из таблицы видно, что ГА с модифицированным кодированием находит точное решение в 1.3 – 2 раза чаще, чем ГА с классическим кодированием.

Табл. 1. Сравнение ГА со стандартным и модифицированным кодированием

№ теста		1	2	3	4	5	6	7	8	9	10
% запусков, в которых было найдено точное решение	классическое кодирование	23,2	41,2	46,6	18,8	35,8	10,8	14,2	9,2	10	22,6
	модифицир. кодирование	43,8	52,2	64,6	27,8	45,6	19,2	19,8	12,6	19,2	31

Приведенные результаты свидетельствуют об эффективности двухуровневого кодирования в смысле повышения качества решения, а также скорости нахождения приемлемого решения.

Заключение

Для рассмотренной задачи реконфигурации ПЛИС был разработан генетический алгоритм решения первого этапа задачи – поиска минимального количества покрывающих прямоугольников. Особенностью алгоритма является двухуровневый способ кодирования особей в ГА. Этот способ позволяет использовать стандартные генетические операторы при соблюдении ограничений задачи. Предложенное решение может быть распространено на случай частичных отказов, когда КЛБ сохраняет частичную работоспособность.

Для реализации алгоритма не требуется дополнительных ресурсов на ПЛИС. Предполагается, что алгоритм выполняется на внешнем универсальном вычислительном устройстве, обладающем достаточной вычислительной мощностью и подключаемом к плате ПЛИС для диагностики и реконфигурации. По этой причине при оценке приближенных алгоритмов их скорость работы менее значима, чем их точность. Анализируя эффективность предложенного генетического алгоритма можно сказать, что его скорость работы принципиально лучше скорости точных переборных алгоритмов, а в

сравнении с приближенными алгоритмами он работает в среднем медленнее, но позволяет найти более точное решение. Применение предложенного двухуровневого кодирования на модельных задачах оптимизации показывает, что генетический алгоритм, использующий стандартные операторы, позволяет найти решение, более близкое к оптимальному, в 70-80% случаев (для задачи поиска кратчайшего цикла).

Таким образом, предложенный алгоритм позволяет увеличить точность решения задачи размещения, что ведет к уменьшению времени реконфигурации и, тем самым, к увеличению коэффициента готовности ПЛИС.

Литература

[Aho, 1987] A.V. Aho, J.E. Hopcroft, J.D. Ullman, Data Structures and Algorithms, Addison-Wesley, USA, 1987

[del Solar, 2006] M.R. del Solar, J.M.S. Perez, J.A.G. Pulido, M.A.V. Rodriguez. Placement and routing of Boolean functions in constrained FPGAs using a distributed genetic algorithm and local search. In: Parallel and Distributed Processing Symposium, IEEE, 2006

[Mitchell, 1999] Mitchell M. An Introduction to Genetic Algorithms. Fifth printing. Cambridge, MA: The MIT Press, 1999

[Tyurin, 1999] S.F. Tyurin. Retention of functional completeness of Boolean functions under "failures" of the arguments. In: Automation and Remote Control, 1999

[Tyurin, 2011] S. Tyurin, V. Kharchenko. Redundant Basises for Critical Systems and Infrastructures: General Approach and Variants of Implementation. In: Proceedings of the 1st Intrenational Workshop on Critical Infrastructures Safety and Security, Kirovograd, Ukraine, 2011

[Tyurin, 2013] S.F. Tyurin, A.V. Grekov, O.A. Gromov, I.S. Ponurovskiy. Adaptable Logical FPGA-Elements. In: Радіоелектронні і комп'ютерні системи, Kharkov, Ukraine, 2013

[Yakimets, 2011] N. Yakimets, V. Kharchenko. Reliable FPGA-Based Systems out of Unreliable Automata: Multi-Version Design Using Genetic Algorithms. In: Design of Digital Systems and Devices, Springer, Berlin, 2011

[Zeidman, 2002] B. Zeidman. Designing with FPGAs and CPLDs, CMP Books, Lawrence, USA, 2002.

[Асанов, 2001] М.О. Асанов, В.А. Баранский, В.В. Расин, Дискретная математика: графы, матроиды, алгоритмы, НИЦ «Регулярная и хаотическая динамика», Ижевск, 2001

[Гладков, 2006] Л.А. Гладков, В.В. Курейчик, В.М. Курейчик. Генетические алгоритмы, М.:ФИЗМАТЛИТ, 2006

[Данилова, 2009] Данилова Е.Ю., Городилов А.Ю. Сравнение генетических алгоритмов на примере задачи коммивояжера // Вестник Пермского ун-та, вып. 3(29), серия «Математика. Механика. Информатика», 2009

[Уваров, 2007] С.С. Уваров. Проектирование реконфигурируемых отказоустойчивых систем на ПЛИС с резервированием на уровне ячеек. // Автоматика и телемеханика, №9, 2007

[Цыбин, 2006] С.А. Цыбин, А.В. Быстрицкий, С.Н. Скуратович. Архитектура отказоустойчивой ПЛИС емкостью свыше 100 тыс. вентилей // Сборник трудов всероссийской научно-технической конференции "Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС)", 2006

Сведения об автора

Алексей Городилов *– старший преподаватель каф. МО ВС, Пермский государственный национальный исследовательский университет, Россия, г. Пермь, 614990, ул. Букирева, д. 15; e-mail: gora830@yandex.ru*

Основные научные интересы: Генетические алгоритмы, эвристические методы, комбинаторные алгоритмы

Two-level genetic algorithm for programmable logic devices reconfiguration

Aleksey Gorodilov

Abstract*: The article considers the problem of programmable logic device reconfiguration after a failure of logic elements. The task is relevant to areas such as space exploration and important industrial facilities management that use highly reliable fault-tolerant systems. The article considers the failures of logic elements in the configurable logic blocks. For optimal reconfiguration logic modules must be placed on a programmable logic device without using faulty logic elements, while minimizing occupied area, the bond lengths and delays. The first step in solving the problem is to choose compactly located logic elements that will be involved in the implementation of the circuit. Compactness is understood as the minimum amount of information necessary to describe the set of logic elements. This task can not be solved exactly and effectively, but can be approximately solved by using various heuristics, including genetic algorithms. This article describes several well-known approaches to the problem and offers an original two-level coding method for constructing a genetic algorithm that takes into account features of the problem and uses standard genetic operators. The proposed algorithm is implemented in practice in the form of a computer program to compare its performance with some other algorithms. Comparison of different heuristic algorithms in terms of accuracy of the solution and estimated speed of the proposed algorithm are given. One can say that the proposed genetic algorithm works on average longer than other heuristic algorithms, but finds a more accurate solution. This leads to a reduction of reconfiguration time and, thereby, increase availability ratio of the programmable logic device.*

Keywords*: programmable logic devices, reconfiguration, reliability, genetic algorithm.*