МИКРОЭЛЕКТРОНИКА, 2019, том 48, № 3, с. 211–223

ИЗБРАННЫЕ МАТЕРИАЛЫ КОНФЕРЕНЦИИ “ПРОБЛЕМЫ РАЗРАБОТКИ ПЕРСПЕКТИВНЫХ МИКРО- И НАНОЭЛЕКТРОННЫХ СИСТЕМ” ПРОЕКТИРОВАНИЕ ИНТЕГРАЛЬНЫХ СХЕМ

УДК 621.3.049.771.14:004.021

МАРШРУТ ТОПОЛОГИЧЕСКОГО СИНТЕЗА ДЛЯ РЕКОНФИГУРИРУЕМЫХ СИСТЕМ НА КРИСТАЛЛЕ СПЕЦИАЛЬНОГО НАЗНАЧЕНИЯ

© 2019 г. С. В. Гаврилов[1, *], Д. А. Железников[1, **], М. А. Заплетина[1], В. М. Хватов[1], Р. Ж. Чочаев[1], В. И. Эннс[2, *]**

[1]Институт проблем проектирования в микроэлектронике Российской академии наук (ИППМ РАН) Советская ул., 3, Зеленоград, Москва, 124365 Россия

[2]Научно-исследовательский институт молекулярной электроники (АО “НИИМЭ”) 1-ый Западный проезд, 12, стр. 1, АО НИИМЭ, Зеленоград, Москва, 124460 Россия

**E-mail: sergey_g@ippm.ru*

***E-mail: zheleznikov_d@ippm.ru*

****E-mail: venns@niime.ru*

Поступила в редакцию 20.11.2018 г.
После доработки 20.11.2018 г.
Принята к публикации 03.12.2018 г.

Описывается маршрут топологического проектирования для реконфигурируемых систем на кристалле, разработанный в ИППМ РАН совместно с АО “НИИМЭ” для схем специального назначения производства ПАО “Микрон”. Разработанный маршрут включает в себя новые подходы к решению топологических задач на таких этапах проектирования, как декомпозиция исходной схемы, размещение логических элементов и трассировка межсоединений. Рассмотренные подходы позволяют ускорить разработку крупных IP-блоков в базисе программируемой логики в условиях совмещения на одном кристалле разнообразных схемотехнических решений, касающихся типов и режимов функционирования коммутационных элементов, а также элементов системы на кристалле.

DOI: 10.1134/S0544126919030050

1. ВВЕДЕНИЕ

В настоящее время реконфигурируемые системы на кристалле (**РСнК**) являются одним из динамично развивающихся направлений цифровой схемотехники. Реконфигурируемые системы на кристалле реализуют интегральные устройства, объединяющие встроенный процессор, реконфигурируемую логику, память и прочие вспомогательные устройства и блоки на одном кристалле. Однокристальное реконфигурируемое или программируемое решение предоставляет гораздо более широкий диапазон системных ресурсов для большей функциональной гибкости и адаптивности, чем традиционные заказные микросхемы класса ASIC, так как допускает оперативное изменение своей внутренней аппаратной структуры.

Для РСнК нет необходимости в проектировании сложно-функциональных блоков (**СФ-блоков**) специального назначения. В своем составе они уже содержат универсальные СФ-блоки, как правило, средней производительности, неоднократно проверенные в работе. За счет этого появляется возможность значительно снизить финансовые затраты на их разработку. Таким образом, изделия класса РСнК обеспечивают комбинацию гибкости проектирования и скорости выхода конечной продукции на рынок без привлечения значительных инвестиций на начальном этапе.

Текущее состояние российской микроэлектроники специального назначения соответствует мировому уровню. Серийность микросхем для таких областей, как авионика, космическая и атомная промышленность крайне низка – сотни и даже десятки микросхем, что выводит на первый план полузаказные схемы на основе БМК и РСнК. Такие свойства как удобство применения и отладки, функциональная полнота, исключение этапов разработки шаблонов в настоящее время приводят к ситуации, когда реконфигурируемые системы становятся наилучшей элементной базой для аппаратуры космического применения. Компания АО “НИИМЭ”, крупнейший российский разработчик интегральных микросхем и RFID-продукции, в рамках программы импортозамещения в 2016 г. начала разработку радиационно-стойких программируемых интегральных

Рис. 1. Структура маршрута топологического проектирования.

микросхем космического применения с технологией КНИ 180 нм.

Для разработки такого класса устройств и решения пользовательских задач на их основе необходимо тесное взаимодействие между аппаратной и программной составляющими маршрута проектирования. Зарубежные производители ПЛИС (Altera, Xilinx, Atmel), а также крупнейшие разработчики в области САПР (Synopsys, Cadence, Mentor Graphics) имеют ряд аппаратных и программных средств для проектирования схем в базисе ПЛИС и РСнК, однако в открытых источниках упомянуты лишь общеизвестные тривиальные методы и подходы. В данном разрезе необходимость компенсировать отставание от западных разработчиков в области программных средств для проектирования схем на РСнК встает особенно остро.

2. МАРШРУТ ТОПОЛОГИЧЕСКОГО ПРОЕКТИРОВАНИЯ

Маршрут топологического синтеза при проектировании схем в базисе реконфигурируемых систем на кристалле в общем виде повторяет этапы классического топологического синтеза заказных схем класса ASIC [1]. Единственным отличием является то, что в результате традиционного заказного проектирования формируется набор масок для проведения фотолитографических процессов, а в случае программируемой логики – загрузочная цифровая последовательность (прошивка) для непосредственного конфигурирования системы (рис. 1).

Предваряющим этапом в маршруте является этап логического синтеза, в ходе которого из регистрового описания схемы на языке Verilog формируется структурное описание в терминах библиотечных элементов РСнК или функциональ-

ных таблиц истинности (Look-Up-Table, LUT). Для решения задачи логического синтеза была осуществлена интеграция с коммерческими программными средствами полузаказного проектирования от компаний Cadence (RTL-Compiler, Genus Synthesis Solution), Synopsys (Design Compiler), а также со свободно распространяемыми средствами логического синтеза и оптимизации схем Yosys [2] и ABC [3].

После получения структурного описания проектируемой схемы следует этап ее декомпозиции на подсхемы в соответствии с архитектурными особенностями РСнК для сокращения размерности задачи последующего этапа размещения элементов. Архитектура РСнК специального назначения может задавать такие ограничения для формируемых подсхем, как их размер, количество внешних и внутренних межсоединений, а также связность с внешними СФ-блоками и блоками ввода-вывода. Качество проведенной декомпозиции имеет важнейшее значение, т.к. от ее результатов в большой мере зависит конечное быстродействие проектируемого устройства.

Следом за декомпозицией исходной схемы наступает этап размещения сформированных подсхем в выделенных областях. Основная задача этапа размещения элементов РСнК заключается в том, чтобы назначить все размещаемые элементы (в том числе СФ-блоки) на легальные позиции таким образом, чтобы оптимизировать некоторую предопределенную целевую функцию, например, общую длину межсоединений. Архитектурные особенности РСнК на данном этапе также накладывают свои ограничения в виде схемотехнических особенностей взаимного расположения элементов для нахождения минимально возможного пути между ними и, соответственно, сокращения задержек распространения сигналов.

Заключительным в маршруте топологического синтеза является этап трассировки межсоединений, в ходе которого элементы проектируемой схемы соединяются друг с другом посредством конфигурирования коммутационных элементов и блоков. Трассировка межсоединений представляет собой сложную комбинаторную задачу оптимизации, особенно для современных РСнК специального назначения в условиях дополнительных ограничений, накладываемых на количество и распределение коммутационных ресурсов на кристалле. Кроме того, российские реконфигурируемые микросхемы в отличие от западных аналогов имеют в своем составе различные типы коммутационных элементов, что приводит к необходимости создания новой графовой коммутационной модели для их описания.

После завершения топологического синтеза остается лишь сформировать загрузочную последовательность для РСнК на основе карты памяти элементной базы. Загрузочная последовательность (или битовый поток, bitstream) – это двоичная последовательность конфигурационных данных, которая осуществляет непосредственное программирование системы путем переключения коммутационных и логических элементов в соответствии с результатами топологического синтеза.

Дальнейшее содержание данной работы организовано следующим образом. В разделе III описывается разработанный алгоритм декомпозиции проектируемых схем. В разделе IV описан подход к иерархическому размещению элементов в базисе РСнК. Раздел V посвящен разработке графовой коммутационной модели и решению задачи трассировки межсоединений на ее основе. И наконец, в разделе VI приведено заключение.

3. ДЕКОМПОЗИЦИЯ

Декомпозиция исходного списка межсоединений на список подсхем является одним из основных этапов проектирования на РСнК. Применение эффективных алгоритмов на данном этапе позволяет как упростить последующий этап размещения элементов, так и повысить быстродействие конечного устройства за счет уменьшения количества используемых трассировочных ресурсов.

Главная задача на этапе декомпозиции – равномерное распределение элементов схемы на заданное количество подсхем с ограничением по количеству элементов в каждой подсхеме. Целью оптимизации на данном этапе является повышение структурной однородности создаваемых подсхем, т.е. уменьшение количества внешних соединений каждой подсхемы с сохранением равномерности распределения элементов.

В настоящее время существующие алгоритмы декомпозиции можно разбить на две большие группы: нисходящие (*top-down*) и восходящие (*bottom-up*). Нисходящие алгоритмы делят исходный список межсоединений на подсхемы “сверху-вниз”, после чего выполняется размещение элементов подсхемы в выделенных областях. К данной группе можно отнести такие известные итерационные алгоритмы как алгоритм Кернигана–Лина [4] и его модификация – алгоритм Федуччи–Маттеуса [5]. Восходящие алгоритмы выполняют объединение элементов исходной схемы “снизу-вверх” в подсхемы – кластеры. К данному семейству алгоритмов можно отнести накопительные алгоритмы кластеризации iRAC [6] и T-VPack [7].

Для решения задачи декомпозиции в маршруте проектирования РСнК с учетом их архитектурных особенностей был разработан и реализован собственный итерационный стохастический алгоритм на основе метода имитации отжига [8].

В основе разработанного подхода лежит идея применения правила Рента [9] в целевой оценочной функции алгоритма, которая выглядит следующим образом:

$$Cost = \sum_{i=1}^{Nc} p_i N_i / N_g, \quad (1)$$

где p_i – экспонента Рента i-ой подсхемы, N_i – количество логических элементов в i-ой подсхеме, N_g – общее количество логических элементов, N_c – общее количество созданных подсхем.

Экспонента Рента является эвристической оценкой сложности межсоединений архитектуры. Значение этой экспоненциальной переменной варьируется от 0 до 1 и для каждой отдельной подсхемы вычисляется по классической формуле:

$$p_i = \frac{\lg(N_{io}/K)}{\lg(B_i)}, \quad (2)$$

где N_{io} – количество внешних терминалов подсхемы, B_i – количество элементов в i-ой подсхеме, K – количество соединений, в среднем приходящихся на один логический элемент.

Кроме целевой оценочной функции метод имитации отжига характеризуют три основные составляющие:

1. начальное решение;
2. график снижения температуры;
3. начальное и конечное значения температуры.

По результатам численных экспериментов для генерации начального решения в разработанном маршруте был выбран алгоритм iRAC с модифицированными параметрами, учитывающими особенности архитектуры РСнК [9]. Для получения начального решения также может быть использован любой другой алгоритм декомпозиции.

Начальная температура вычисляется следующим образом:

$$T_{\text{initial}} = 10000 \times Cost_{\text{initial}}, \quad (3)$$

где $Cost_{\text{initial}}$ – значение целевой оценочной функции после генерации начального решения. Конечная температура: $T_{\text{end}} = 0.00001$. Количество итераций алгоритма L при температуре T: $L = 100 \times N_g$.

График охлаждения рассчитывается по классической формуле [11]: $T_{\text{new}} = aT_{\text{old}}$, где T_{old} – предыдущее значение температуры, а α является параметром, который зависит от доли принятых решений (R_{accepted}) при температуре T_{old}, как показано в табл. 1.

Генерация нового решения на каждой итерации алгоритма выполняется либо путем взаимной перестановки двух случайных логических элементов из разных подсхем, либо одиночного переноса логического элемента. Равномерность распределения элементов по подсхемам при одиночных переносах сохраняется благодаря ограничению на размеры подсхем, заданному следующим образом:

$$\begin{aligned} MinSize &= AverageSize \times \left(1 - \frac{\text{MaxDeviation}}{100}\right), \\ MaxSize &= AverageSize \times \left(1 + \frac{\text{MaxDeviation}}{100}\right), \end{aligned} \quad (4)$$

где *MinSize* – минимально возможный размер подсхемы, *MaxSize* – максимально возможный, *AverageSize* – средний размер кластера, задаваемый пользователем. С помощью параметра *MaxDeviation* задается максимальное отклонение размера подсхемы от среднего значения.

Таким образом, итерационная минимизация целевой оценочной функции ведет к увеличению структурной однородности создаваемых подсхем и, соответственно, приводит к уменьшению количества требуемых трассировочных ресурсов, а также общему увеличению быстродействия проектируемой схемы.

Таблица 1. График снижения температуры

$R_{accepted}$	α
От 0.96 до 1	0.5
От 0.8 до 0.96	0.9
От 0.15 до 0.8	0.95
От 0 до 0.15	0.8

4. РАЗМЕЩЕНИЕ

За этапом декомпозиции в маршруте топологического проектирования следует этап размещения элементов. От результатов, полученных на данном этапе, напрямую зависят результаты трассировки и основные характеристики спроектированной цифровой схемы. При этом получение качественного размещения – непростая задача, т.к. емкость современных РСнК превышает десятки тысяч конфигурируемых ЛЭ.

Среди современных подходов можно выделить три основные группы алгоритмов размещения [12]:

1. дихотомические алгоритмы;

2. аналитические алгоритмы;

3. эвристические методы.

В алгоритмах дихотомического размещения область размещения и сама схема последовательно разбивается на части, а затем на каждую полученную подобласть назначается определенная подсхема. Дихотомические алгоритмы, как правило, направлены на минимизацию сечения, что приводит к уменьшению количества требуемых для трассировки ресурсов. Данные алгоритмы хорошо подходят для ПЛИС с симметричной иерархической архитектурой. Например, к данному семейству алгоритмов относится алгоритм ALTOR [13], адаптированный для таких ПЛИС, как Altera Apex 20 K [14]. Основными недостатками дихотомических подходов являются низкая устойчивость к малым изменениям на входе и сложность учета нескольких разнородных критериев.

Аналитические алгоритмы размещения появились относительно недавно. Задача размещения в данном случае представляется в виде системы линейных уравнений и решается с помощью численных методов. Полученное решение не является легальным с точки зрения перекрытия элементов, поэтому данные алгоритмы содержат механизмы его легализации. К аналитическим алгоритмам относятся алгоритмы StarPlace [15], RippleFPGA [16] и HeAP [17]. Основным преимуществом данных алгоритмов является высокая скорость работы, что в условиях стремительного роста размерности РСнК является одним из ключевых параметров. Однако необходимость в качественном алгоритме легализации и общая сложность реализации сводит на нет их преимущества для схем среднего и небольшого объема.

Другой группой популярных методов являются эвристические алгоритмы. Эвристические подходы используют различные приемы, позволяющие при решении задачи размещения избегать “застревания” в локальных минимумах целевой функции при поиске глобального. К данной группе алгоритмов относятся метод имитации отжига [8], генетические алгоритмы [18] и другие. Наиболее популярным как в коммерческих, так и академических приложениях является метод имитации отжига. К известным реализациям данного метода можно отнести VPR [11] и алгоритм размещения, реализованный в САПР Altera Quartus [19].

Для реализации этапа размещения элементов в базисе РСнК был разработан иерархический подход на основе метода имитации отжига. Данный метод был адаптирован для размещения цифровых схем в базисе семейства РСнК, разрабатываемого АО “НИИМЭ” и условно именуемого далее “А01”.

Семейство РСнК “А01” имеет островную иерархическую архитектуру [12] (рис. 2), сочетающую в себе программируемую логику и набор СФ-блоков специального назначения (далее макроблоки) таких, как 16-тиразрядный умножитель, блок фазовой автоподстройки частоты, блок низковольтной дифференциальной передачи сигналов и блоки встроенной памяти с произвольной выборкой. Всего программируемая часть содержит более 25 тысяч логических элементов (**ЛЭ**), объединенных в группы (**ГЛЭ**) по 256 элементов. Логический элемент состоит из двух логических ячеек (**ЛЯ**), каждая из которых может реализовывать логическую функцию до трех переменных или D-триггер. С помощью быстрых локальных шин ЛЭ может быть соединен с соседними ЛЭ по вертикали и горизонтали (рис. 3). Из рисунка видно, что связи между ЛЭ имеют направленный характер, что необходимо учитывать на этапе размещения.

С учетом двухуровневой иерархической структуры РСнК “А01” алгоритм размещения был разбит на два этапа. На первом этапе выполняется размещение ГЛЭ, на втором - логических элементов внутри ГЛЭ.

А. Этап размещения ГЛЭ

Для генерации начального размещения групп логических элементов применяется силовой алгоритм, после чего выполняется оптимизация полученного размещения с помощью алгоритма имитации отжига. На каждой итерации алгоритма выполняется перестановка местами двух случайных ГЛЭ или перенос одной ГЛЭ на новую позицию, после чего производится оценка значения целевой функции.

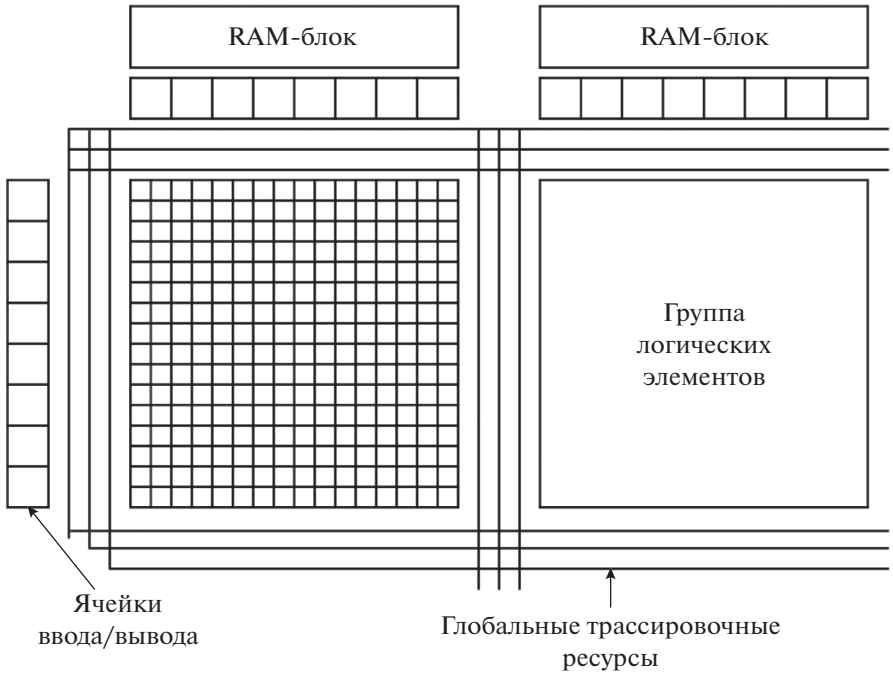

Рис. 2. Архитектура РСнК “A01”.

Разработанная целевая оценочная функция имеет следующий вид:

$$Cost = \sum_{k \in Labs} \left(\sum_{i \in Inputs} InputCost_k^i + \sum_{j \in Outputs} OutputCost_k^j \right), \quad (5)$$

где $InputCost_k^i$ и $OutputCost_k^j$ – “входная” и “выходная” составляющие целевой функции, *Labs* – список всех ГЛЭ; *Inputs*, *Outputs* – списки входов и выходов *k*-ой ГЛЭ.

“Входная” составляющая целевой функции имеет вид:

$$InputCost_k^i = 5\alpha\left(|x_k - x_i| + |y_k - y_i|\right), \quad (6)$$

где (x_k, y_k) – координаты центра *k*-ой ГЛЭ, (x_i, y_i) – координаты источника *i*-го сигнала, α – коэффициент, равный 4, если среди терминалов цепи есть ячейка ввода–вывода или макроблок, и равный 1 во всех остальных случаях.

“Выходная” составляющая вычисляется следующим образом:

$$OutputCost_k^j = 5\beta\left(|x_k - x_t| + |y_k - y_t|\right), \quad (7)$$

где *t* – терминал *p*-ого выхода *l*-ой ГЛЭ, (x_t, y_t) – координаты блока, подключенного к *t*-ому терминалу, β – коэффициент равный 3, если среди терминалов цепи есть макроблок или ячейка ввода-вывода, и равный 1 во всех остальных случаях.

Для вычисления начального и конечного значений температуры, количества итераций используются методы, описанные в работе [10].

График понижения температуры имеет следующий вид:

$$T_{new} = \alpha T_{old}, \quad (8)$$

где T_{old} – предыдущее значение температуры, а α является параметром, который зависит от доли принятых решений ($R_{accepted}$) при температуре T_{old} (табл. 1).

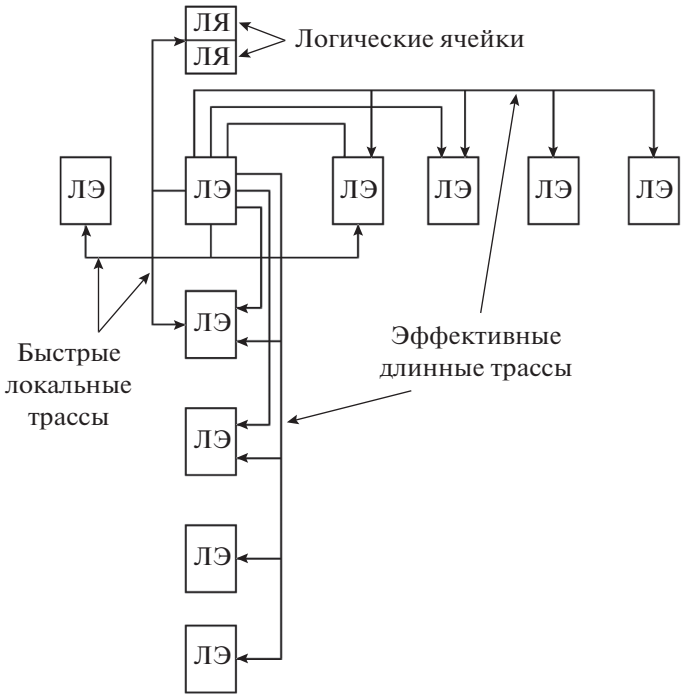

Рис. 3. Локальные трассировочные ресурсы.

Б. Этап размещения ЛЭ внутри ГЛЭ

На стадии инициализации алгоритма размещения логических элементов внутри ГЛЭ определяется порядок ГЛЭ, в котором будет выполняться размещение. Сортировка ГЛЭ выполняется таким образом, чтобы первыми обрабатывались ГЛЭ, расположенные в верхних рядах кристалла. В пределах одного ряда ГЛЭ сортируются по возрастанию j-ой координаты, т.е. слева направо.

Когда порядок ГЛЭ определен, начинается этап начального размещения ЛЭ. На первом шаге размещаются ЛЭ, выход которых играет роль локального тактового сигнала. На втором шаге оставшиеся ЛЭ сортируются по убыванию связности с макроблоками и ячейками ввода-вывода и в соответствии с полученным порядком размещаются последовательно вдоль границы ГЛЭ, что приводит к тому, что элементы, не имеющие внешних соединений, оказываются на внутренних позициях в ГЛЭ.

После завершения этапа начального размещения выполняется оптимизация с помощью адаптированного алгоритма имитации отжига. Разработанная целевая функция имеет следующий вид:

$$Cost = \sum_{k \in LEs} \left(\sum_{i \in Inputs} InputCost_k^i + \sum_{i \in Outputs} OutputCost_k^j \right), \quad (9)$$

где LEs – список ЛЭ; $Inputs$, $Outputs$ – списки входов и выходов k-ого ЛЭ соответственно; $InputCost_k^i$ и $OutputCost_k^j$ – “входная” и “выходная” составляющие целевой функции.

Значение “входной” составляющей целевой функции зависит от типа ячейки, которая является источником сигнала i-го входа ЛЭ, и ее расположения. Если ячейка-источник является макроблоком или ячейкой ввода-вывода и она расположена в верхней или нижней части кристалла, то “входная” составляющая имеет вид:

$$InputCost_k^i = \alpha |x_k - x_i|, \quad (10)$$

где x_k – координата ЛЭ по оси x, x_i – координата по оси x ячейки-источника.

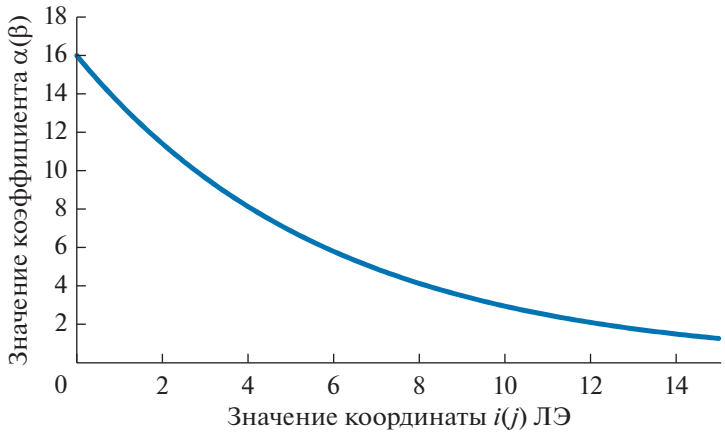

Рис. 4. Зависимость значения штрафных коэффициентов α и β от координат ЛЭ.

Если ячейка расположена в левой или правой части кристалла, то:

$$InputCost_k^i = \beta|y_k - y_i|, \tag{11}$$

где y_k – координата ЛЭ по оси y, y_i – координата по оси y ячейки-источника.

Штрафные коэффициенты α и β экспоненциально зависят от расположения ЛЭ в ГЛЭ и вычисляются как (рис. 4):

$$\begin{aligned} \alpha &= 16e^{-0.168986LE_k^j}, \\ \beta &= 16e^{-0.168986LE_k^i} \end{aligned}, \tag{12}$$

где LE_k^i – координата по ik-ого ЛЭ, LE_k^j – координата по jk-ого ЛЭ.

Если источником сигнала i-го входа ЛЭ является другой логический элемент, то значение "входной" составляющей целевой функции имеет следующий вид:

$$InputCost_k^i = 5\gamma(|x_k - x_i| + |y_k - y_i|), \tag{13}$$

где (x_k, y_k) – координаты ЛЭ по осям x и y, (x_i, y_i) – координаты по осям x и y ЛЭ-источника.

Параметр γ зависит от взаимного расположения логических элементов. Пусть $\Delta_i = D^i - LE_k^i$ и $\Delta_j = D^j - LE_k^j$. Тогда:

$$y = \begin{cases} 0.1|\Delta_i + \Delta_j|, \text{ если } \Delta_i = 0 \text{ и } \Delta_j \in [-4; 1] \text{ или } \Delta_j = 0 \text{ и } \Delta_i \in [-1; 4] \\ 1, \text{ если } \Delta_i \geq 0 \text{ и } \Delta_j \leq 0 \\ 2, \text{ во всех остальных случаях} \end{cases}. \tag{14}$$

"Выходная" составляющая целевой функции имеет следующий вид:

$$OutputCost_k^j = \sum_{t=1}^{j} a_t OutCostIO_k^t + (1 - a_t)OutCostLE_k^t, \tag{15}$$

где t – терминал j-ого выхода, $OutCostIO_k^t$ – значение целевой функции, если $a_t = 1$, $OutCostLE_k^t$ – если $a_t = 0$.

Коэффициент a_t зависит от того, какой ячейке принадлежит терминал выходной цепи t:

$$a_t = \begin{cases} 0, \text{ если } t \in LE \\ 1, \text{ если } t \notin LE \end{cases}. \tag{16}$$

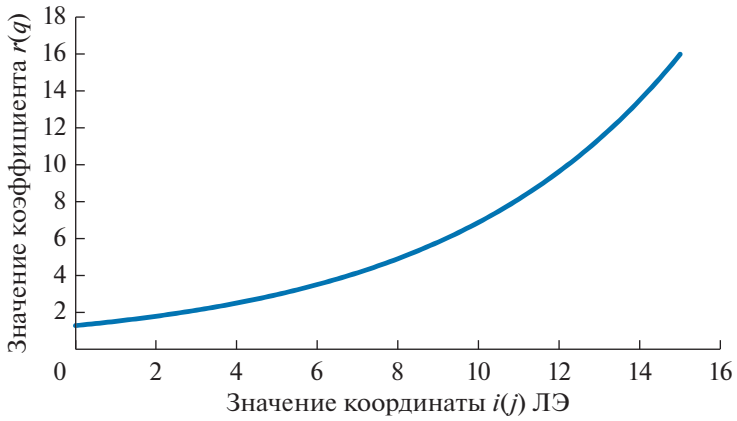

Рис. 5. Зависимость значения штрафного коэффициента *r* от *j*-ой координаты ЛЭ.

Значение *OutCostIO* зависит от того, где находится ячейка, связанная с терминалом *tj*-ого выхода ЛЭ. Если данная ячейка расположена в верхней или нижней части кристалла, то:

$$OutCostLE_k^t = r|x_t - x_k|, \tag{17}$$

где x_k – координата ЛЭ по оси x, x_t – координата ячейки по оси x.

Когда ячейка расположена в правой или левой части кристалла, то:

$$OutCostLE_k^t = q|y_t - y_k|, \tag{18}$$

где y_k – координата ЛЭ по оси y, y_t – координата ячейки по оси y.

Штрафные коэффициенты r и q экспоненциально зависят от расположения ЛЭ в ГЛЭ и вычисляются как (см. рис. 5):

$$\begin{aligned} r &= 1.26871e^{0.168968LE_k^j} \\ q &= 1.26871e^{0.168968LE_k^i}. \end{aligned} \tag{19}$$

Если ячейка, связанная с терминалом *tj*-ого выхода ЛЭ, является обычным ЛЭ, то:

$$OutCostLE_k^t = 5v(|x_t - x_k| + |y_t - y_k|), \tag{20}$$

где v – коэффициент, который зависит от относительного расположения k-го ЛЭ и связанной с ней выходной ячейки-ЛЭ.

Если в ЛЭ используются обе логические ячейки, то:

$$v = 5\gamma, \text{ если } \Delta_j = 0 \text{ и } \Delta_i \in [-1; 4]. \tag{21}$$

Если источником сигнала выходной цепи является вторая ЛЯ в ЛЭ, то:

$$v = 5\gamma, \text{ если } \Delta_i = 0 \text{ и } \Delta_j \in [-4; 1]. \tag{22}$$

Во всех остальных случаях v рассчитывается так же, как и коэффициент γ.

Начальная и конечная температура, количество итераций на каждом шаге и график понижения температуры не отличаются от реализации алгоритма для ГЛЭ, описанного ранее. На каждой итерации разработанного алгоритма выполняется либо перестановка двух случайных ЛЭ, либо перемещение ЛЭ на свободную позицию. При этом область перестановки в зависимости от ЛЭ может различаться. Например, ЛЭ, выход которых является локальным тактовым сигналом, могут перемещаться только в пределах нижней строки ГЛЭ, а ЛЭ, связанные с ячейками ввода-вывода или макроблоками перемещаются только вдоль крайних рядов ГЛЭ.

Применение разработанных алгоритмов на практике показывает, что они позволяют получать качественное размещение на РСнК и, в частности, на семействе “А01”. Благодаря введению

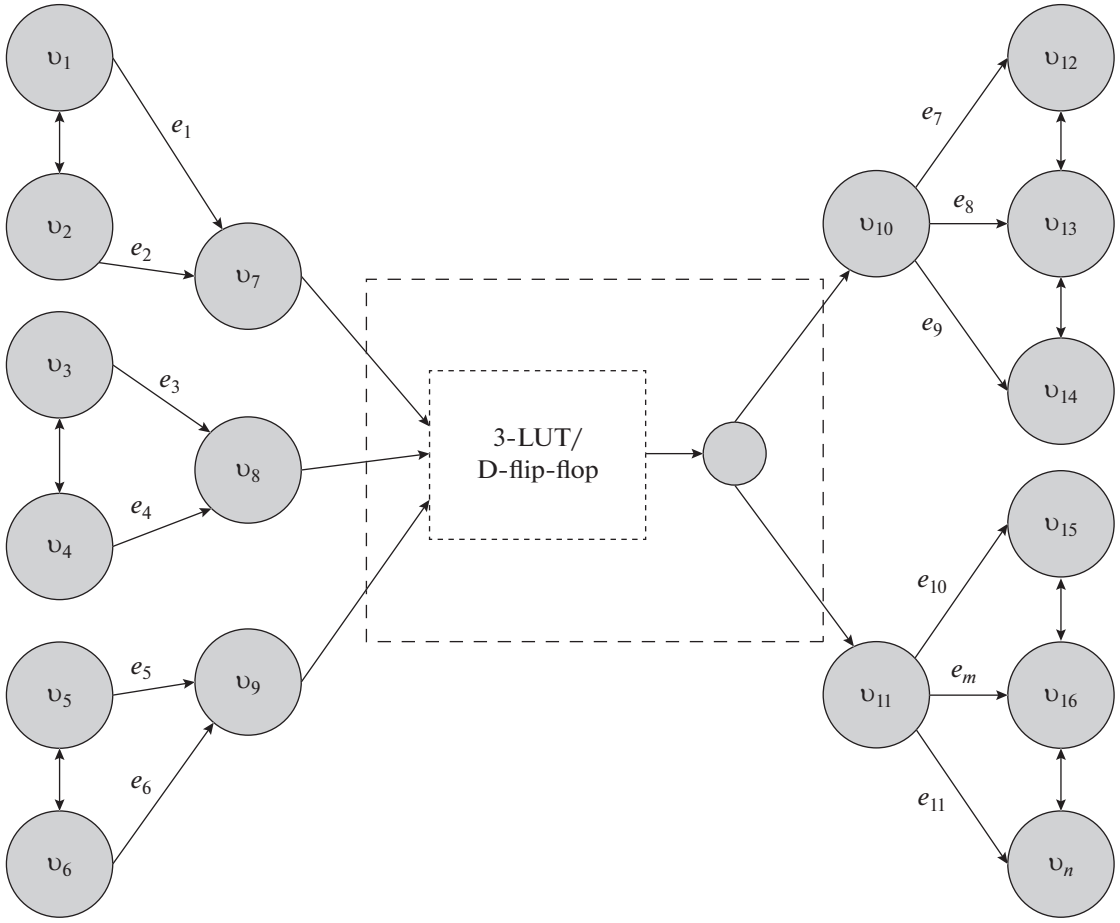

Рис. 6. Пример смешанного графа для трассировки межсоединений (3-LUT – функциональная таблица истинности, D-flip-flop – триггер).

штрафных функций достигается экономия ограниченных ресурсов для коммутации, что также повышает трассируемость цифровых схем на РСнК.

5. ТРАССИРОВКА МЕЖСОЕДИНЕНИЙ

Разрабатываемые АО “НИИМЭ” РСнК специального назначения содержат ряд схемотехнических решений, не имеющих аналогов за рубежом. Одним из таких решений является использование нестандартных для классических ПЛИС типов коммутационных элементов. Помимо стандартных переключателей в виде n-канального МОП транзистора, коммутационные ресурсы программируемой части РСнК содержат дополнительные логические элементы, такие как мультиплексоры, инверторы, многоразрядные усиленные буферы и иные. Сочетание ключей условного и безусловного включения, одно- и двунаправленных, с инверсией и без инверсии, а также с усилением сигнала вызывает трудности в применении классических алгоритмов для трассировки межсоединений и создает необходимость в их адаптации и разработке новых оригинальных решений.

Центральной моделью, представляющей трассировочные ресурсы РСнК, является конечный взвешенный гиперграф $G = \{V, E\}$, где множество $V = \{v_1, \ldots, v_n\}$ соответствуют полному набору сигналов (электрических узлов) схемы, а множество $E = \{e_1, \ldots, e_m\}$ – совокупности всех конфигурируемых трассировочных элементов между ними. В классическом представлении каждой вершине графа ставится в соответствие фиксированный вес $w(v_i)$, $v_i \in V$, а каждому ребру – вес $w(e_j)$, $e_j \in E$, отражающие относительную стоимость прохождения сигнала (задержку) в соответствии с емкостной нагрузкой электрических узлов и схемотехническими особенностями конкретных трассировочных элементов. Другие параметры разнятся в зависимости от рассматриваемого этапа трассировки и будут подробно рассмотрены далее.

В связи с архитектурными особенностями рассматриваемого семейства РСнК, выражающимися, в частности, в широком разнообразии комму-

тационных элементов, возникла необходимость в разработке оригинальной графовой коммутационной модели для описания доступных трассировочных ресурсов [20]. Ее первой отличительной особенностью является сочетание двух типов соединений пар вершин графа трассировки: ориентированных и неориентированных ребер, возникших из потребности в учете направления прохождения сигнала через соответствующий коммутационный элемент (например, инвертор, мультиплексор и пр.). Таким образом, граф из классического ненаправленного становится смешанным (рис. 6).

Каждому ребру смешанного графа также сопоставляется логическая функция прохождения сигнала, позволяющая помимо его направления учесть случай инвертирования сигнала на информационных входах логических элементов, описать влияние управляющего сигнала на коммутационный элемент или их группу, а также корректно описать элементы, коммутируемые по нагрузке (усилению сигнала). На основе разработанной математической модели был реализован программный интерфейс на языке Tcl [20], позволяющий наиболее полно передать данные об имеющихся коммутационных ресурсах в процедуру трассировки.

Анализ эффективности существующих алгоритмов, адаптированных для задачи трассировки РСнК, позволил выработать единый подход, заключающийся в совмещении двух методик: модифицированного алгоритма A* вкупе с механизмом разрыва и перетрассировки [21, 22] и адаптированного алгоритма PathFinder [23, 24], применяемых на базе списка межсоединений, упорядоченного по типам и приоритетам.

Выстраивание списка трассировки цепей возможно по нескольким критериям, например, присвоение более высокого приоритета межсоединениям с наибольшим суммарным числом узлов источников и приемников сигнала или сортировка списка по размеру ограничивающего прямоугольника цепи. В качестве оптимального был выбран способ построения списка трассировки по смешанному критерию. Первыми в список трассировки попадают цепи синхронизации и управляющие сигналы, затем путем оценки разветвленности каждой цепи по числу терминалов выстраивается список оставшихся межсоединений. Кроме автоматического режима, предусмотрено внесение изменений в построенный список по требованию, в частности, существует возможность ручного определения порядка трассируемых цепей.

После упорядочивания полученный список межсоединений передается на процедуру трассировки, состоящую из двух стадий. На первой стадии с помощью модифицированного алгоритма A* трассируются выбранные пользователем, управляющие цепи и цепи синхросигналов. С каждой вершиной графа v_i ассоциируется изменяемая величина $w(v_i)$, равная стоимости прохождения сигнала через v_i и рассчитываемая следующим образом:

$$w(v_i) = g(v_i) + \alpha h(v_i), \tag{23}$$

где $g(v_i)$ – вес минимального пути до вершины v_i, α – единичный (по умолчанию) весовой коэффициент, $h(v_i)$ – приближенная оценка оставшегося пути, получаемая с учетом результатов обратного рекурсивного прохода волнового алгоритма от приемников сигнала до текущей вершины и вычисляемая по формуле:

$$h(v_i) = \min_{\substack{v_j, \exists e_{ij} \in E, \\ e_{ij} = \{v_i, v_j\}}} (h(v_j) + 1). \tag{24}$$

В основу реализации алгоритма положено использование очереди с приоритетами, выполненной в виде двоичной кучи (binary heap), которая представляет собой частично сортированное полное двоичное дерево. Эта структура поддерживает операции просмотра минимального (корневого) элемента за время O(1), а также удаление, обновление ключа вершины или вставку нового ключа за время O(lg(n)), где n – число вершин в куче. Вычислительная эффективность операций над двоичной кучей не является наилучшей среди существующих структур данных, однако простота программной реализации и возможность решения задачи трассировки межсоединений за полиномиальное время оправдывают ее выбор.

Для повышения процента трассируемости модифицированный алгоритм A* был дополнен механизмом разрыва и перетрассировки, позволяющим при помощи, как правило, нескольких детерминированных перестановок в списке высокоприоритетных межсоединений разрешить некоторые конфликты, возникающие при перегрузке (разделении) трассировочных ресурсов несколькими цепями одновременно. Из всех рассмотренных в работе [22] вариантов по соотношению между итоговым приростом процента разведенных межсоединений и затраченного на перестановки времени была выбрана модификация, основанная на перемещении конфликтующей цепи в середину приоритетного списка ранее разведенных.

На втором этапе процедуры трассировки используется адаптация к архитектуре семейства РСнК классического эвристического алгоритма PathFinder на смешанном графе трассировки, представленном выше. На каждом шаге его работы выполняется трассировка полного списка межсоединений, исключая высокоприоритетные, разведенные на первой стадии. Трассировка каждой отдельно взятой цепи выполняется классическим поиском в ширину на графе с учетом весовых характеристик каждого элемента. За исключением последней итерации, в ходе работы алгоритма некоторые вершины графа оказываются перегруженными, и каждая последующая итерация призвана сократить объем разделяемых трассировочных ресурсов до тех пор, пока существуют доступные для использования незанятые коммутационные элементы.

На этом этапе каждой вершине смешанного гиперграфа назначается в соответствие ряд постоянных и изменяемых величин: $w(v_i) = \text{const}$ – собственный вес вершины, имеющий смысл значения внутренней задержки i-го узла схемы; $h(v_i)$ – вариативный параметр, пропорциональный числу цепей, перегружающих вершину на предыдущей итерации алгоритма; $p(v_i)$ – коэффициент, отслеживающий число цепей, разделивших вершину, и номер итерации от начала работы алгоритма (позволяет с ходом времени исключить из рассмотрения массово разделяемые вершины, поскольку стоимость прохождения сигнала через них будет значительно выше, чем через менее нагруженные); $c(v_i)$ – обновляемый расчетный вес вершины. В процессе поиска кратчайшего пути от источника до приемника сигнала для каждой вершины вычисляется стоимость прохождения сигнала:

$$\text{Cost}(v_i) = w_h^i + c(v_i) + w(e_i), \tag{25}$$

где w_h^i – накопленная стоимость пути до текущего узла, а $w(e_i)$ – весовая характеристика ребра, приведшего к вершине.

Следует упомянуть о четырех основных настроечных параметрах в применяемой адаптации алгоритма PathFinder: VP – коэффициент, отвечающий за влияние накопленного и собственного базового веса вершины смешанного графа трассировки; VH – коэффициент учета всех цепей, разведенных через текущую вершину; $PATHW$ – ограничивающая суммарная накопленная стоимость прохождения сигнала через вершину для межсоединения, имеющая смысл максимальной задержки распространения сигнала; $PATHL$ – максимальная длина межсоединения в количестве пройденных трассировочных элементов. Последние две величины во многом определяют качество находимых алгоритмом путей, а также итоговую длительность процедуры трассировки.

Для каждой цепи на i-й итерации происходит обновление весов всех вершин смешанного гиперграфа, попавших в ее дерево трассировки, по следующим правилам:

$$\begin{aligned} p(v_i) &= 1 + (i-1)VPn(v_i) \\ c(v_i) &= p(v_i)(w(v_i) + h(v_i)), \end{aligned} \tag{26}$$

где $n(v_i)$ – число цепей, перегружающих вершину.

Кроме того, в конце каждой итерации происходит обновление весов всех вершин графа, причем для перегруженных вершин проводится пересчет параметра $h(v_i)$ в соответствии со значением коэффициента VH и текущим числом разделяющих их цепей.

Для кристаллов, не обладающих разнообразием конфигураций коммутационных элементов, подбора значений перечисленных коэффициентов достаточно для качественной отладки алгоритма, в то время как для РСнК с широким рядом трассировочных ключей необходим дополнительный подбор базовых собственных весов для каждого из них. Элементарным ориентиром для их определения может являться степень схемотехнической сложности того или иного коммутационного элемента, но в строгом смысле в их основу должны быть положены данные, полученные при характеризации и оценке быстродействия.

Сочетание двух методик: модифицированного алгоритма A* с механизмом разрыва и перетрассировки для критически важных межсоединений и адаптации PathFinder для менее важных цепей – позволяет найти компромисс между скоростью трассировки, степенью разводимости и итоговыми временными характеристиками межсоединений.

6. ЗАКЛЮЧЕНИЕ

В процессе создания собственных инструментальных средств проектирования полузаказных схем на основе РСнК специального назначения были разработаны новые подходы к решению задач декомпозиции, размещения элементов и трассировки межсоединений, позволяющие избежать полной зависимости от проприетарных продуктов и повышающие адаптивность к сложным пользовательским сценариям.

В ходе разработки были предложены оригинальные компоновки и адаптации классических методов решения задач топологического синтеза, учитывающие архитектурные особенности реконфигурируемых систем на кристалле. Представленные методы показали свою эффективность в решении задач на различных этапах маршрута топологического проектирования РСнК специального назначения.

СПИСОК ЛИТЕРАТУРЫ

1. *Гаврилов С.В.* Методы анализа логических корреляций для САПР цифровых КМОП СБИС // М.: Техносфера. 2011. 136 с.
2. Yosys Open Synthesis Suite. Version 0.8. // http://www.clifford.at/yosys/.
3. ABC: A System for Sequential Synthesis and Verification. Berkeley Logic Synthesis and Verification Group. // https://people.eecs.berkeley.edu/~alanmi/abc/.
4. *Kernighan B.W., Lin S.* An efficient heuristic procedure for partitioning graphs // The Bell Sys. Technical J. 1970. V. 49. № 2. P. 291–307.
5. *Fiduccia C.M., Mattheyses R.M.* A Linear-Time Heuristic for Improving Network Partitions // 19th DAC. Las Vegas. NV. USA. 1982. P. 175–181.
6. *Singh A.M., Marek-Sadowska M.* Efficient circuit clustering for Area and Power Reduction in FPGAs // ACM Trans. Des. Autom. Electron. Syst. 2002. V. 7. № 4. P. 643–663.
7. *Marquardt A., Betz V., Rose J.* Using Cluster-Based Logic Blocks to Improve FPGA Speed and Density // ACM Symp. on FPGAs. 1999. P. 37–46.
8. *Kirkpatrick S., Gelatt C.D., Jr., Vecchi M.P.* Optimization by Simulated Annealing // Science. 1983. V. 220. P. 671–680.
9. *Landman B.S., Russo R.L.* On a Pin Versus Block Relationship For Partitions of Logic Graphs // IEEE Transactions on Computers. 1971. V. C-20. № 12. P. 1469–1479.
10. *Гаврилов С.В., Железников Д.А., Чочаев Р.Ж., Хватов В.М.* Алгоритм декомпозиции на основе метода имитации отжига для реконфигурируемых систем на кристалле // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС-2018). 2018. Вып. I. С. 199–204.
11. *Betz V., Rose J.* VPR: A new packing, placement and routing tool for FPGA research // Proc. of the 7th Int. Workshop on Field–Programmable Logic and Applications. 1997. P. 213–222.
12. Reconfigurable computing: the theory and practice of FPGA–based computation // Eds. Hauck S., Dehon A. Morgan Kaufmann Publishers Inc. San Francisco. 2007. 944 p.
13. *Rose J., Snelgrove W., Vranesic Z.* ALTOR: An automatic standard cell layout program // Proc. of the Canadian Conference on VLSI. 1985. P. 168–173.
14. *Hutton M., Adibsamii K., Leaver A.* Adaptive delay estimation for partitioning driven PLD placement // IEEE Trans. on VLSI. 2003. V. 11. № 1. P. 60–63.
15. *Xu M., Grewal G., Areibi S.* StarPlace: A new analytic method for FPGA placement // Integration, the VLSI J. 2011. V. 44. Is. 3. P. 192–204.
16. *Pui Ch.-W., Chen G.* RippleFPGA: A routability-driven placement for large-scale heterogeneous FPGAs // 2016 IEEE/ACM Int. Conf. on CAD. 2016. P. 1–8.
17. *Gort M., Anderson J.H.* Analytical placement for heterogeneous FPGAs // 22nd Int. Conf. on Field Programmable Logic and Applications (FPL). 2012. P. 143–150.
18. *Rabozzi M., Durelli G.C., Miele A., Lillis J., Santambrogio M.D.* Floorplanning Automation for Partial-Reconfigurable FPGAs via Feasible Placements Generation // IEEE Trans. on VLSI Systems. 2017. V. 25. Is. 1. P. 151–164.
19. *Ludwin A., Betz V.* Efficient and Deterministic Parallel Placement for FPGAs // ACM Trans. Des. Autom. Electron. Syst. 2011. V. 16. № 3. P. 22.1–22.23.
20. *Гаврилов С.В., Железников Д.А., Хватов В.М.* Решение задач трассировки межсоединений с ресинтезом для реконфигурируемых систем на кристалле // Изв. вузов. Электроника. 2017. Т. 22. № 3. С. 266–275.
21. *Dees W., Smith R.* Performance of Interconnection Rip-Up and Reroute Strategies // Proc. of 18th DAC. 1981. P. 382–390.
22. *Железников Д.А., Заплетина М.А., Хватов В.М.* Исследование механизма разрыва и перетрассировки на этапе топологического синтеза в базисе реконфигурируемых систем на кристалле // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС-2018). 2018. Вып. I. С. 188–192.
23. *McMurchie L., Ebeling C.* PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs // 3rd Int. ACM Symp. on FPGAs. Napa Valley. CA. USA. 1995. P. 111–117.
24. *Железников Д.А., Заплетина М.А., Хватов В.М.* Решение задачи трассировки межсоединений для реконфигурируемых систем на кристалле с различными типами коммутационных элементов // Электронная техника. Сер. 3. Микроэлектроника. 2019. Вып. 4(172). С. 31–36.