УДК 621.3.049.771.14 DOI: 10.31114/2078-7707-2020-1-2-7

Алгоритм размещения с оптимизацией быстродействия на основе матриц задержек для реконфигурируемых систем на кристалле

П.И. Фролова, Р.Ж. Чочаев, Г.А. Иванова, С.В. Гаврилов

Институт проблем проектирования микроэлектроники РАН, г. Москва,

frolova_p@ippm.ru, chochaev_r@ippm.ru, ivanova_g@ippm.ru, s.g@ippm.ru

***Аннотация* — Реконфигурируемая система на кристалле (РСнК) представляет собой устройство, в котором на одном кристалле расположены конфигурируемые логические блоки и жесткие IP ядра. При работе с РСнК этап размещения элементов является критически важным и трудоёмким этапом в маршруте проектирования. Результаты этапа размещения сильно влияют на трассируемость и быстродействие схем. Поэтому при разработке высокоскоростных цифровых схем на базе РСнК важно использовать эффективные алгоритмы размещения, которые учитывают задержки. В данной статье мы представляем новый алгоритм размещения с оптимизацией быстродействия на основе метода имитации отжига для РСнК островного типа. Была разработана новая функция оценки, основанная на модели полупериметра охватывающего прямоугольника и новой модели задержек. Для оценки задержек в глобальных и локальных цепях используются матрицы, в которых хранятся значения задержек. Разработанный алгоритм реализован и протестирован на наборах тестовых схем ISCAS-85 и ISCAS-89. По сравнению с алгоритмом размещения, оптимизирующим длину межсоединений, результаты применения представленного алгоритма обеспечивают меньшие значения задержек, что приводит к повышению быстродействия схем на РСнК.**

***Ключевые слова* — ПЛИС, реконфигурируемая система на кристалле (РСнК), средства автоматизации проектирования (САПР).**

I. Введение

Реконфигурируемая система на кристалле в своем составе содержит программируемую логику, память, центральный процессор и различные IP-блоки. Для проектирования высокоскоростных систем на основе РСнК возникает необходимость в применении эффективных алгоритмов оптимизации быстродействия схем на этапе размещения. Этап размещения в маршруте проектирования РСнК является одним из самых сложных и трудоемких [1]. На данном этапе определяется легальное расположение элементов из нетлиста на кристалле с учетом особенностей архитектуры РСнК. Существующие методы размещения можно разделить на несколько групп, в зависимости от функции оценки: длина межсоединений, быстродействие, перегруженность и т.д. Алгоритмы размещения с оптимизацией быстродействия используют данные, полученные после статического временного анализа, для оптимизации задержек.

Статический временной анализ [2, 3, 4, 5] позволяет оценить быстродействие схемы и выделить критические пути. Логическая схема представляется в виде ациклического направленного графа **G**(**V**, **E**), где **V** – множество вершин графа (терминалы элементов схемы), **E** – множество ребер графа (связи между терминалами цепей). Каждое ребро графа характеризуется задержкой *d*(*e*), *e* ∈ **E**. Чтобы вычислить задержку необходимо вычислить время прибытия сигнала (*arrival time*) по следующей формуле:

$$AT(v) = \max_{\forall u \in fanin(v)} AT(u) + d(u,v), \qquad (1)$$

где *AT(v)* – время прибытия сигнала для вершины *v*, т.е. время, за которое сигнал распространяется от первичных входов до данной вершины; *d(u, v)* – задержка ребра *(u, v)*. Тогда задержка схемы D_{max} будет вычисляться следующим образом:

$$D_{\max} = \max_{\forall v \in PO} AT(v), \qquad (2)$$

где *PO* - первичные выходы схемы.

Существующие методы оптимизации быстродействия можно разделить на 3 большие группы [5]: *на основе соединений* (*net-based*), *на основе путей* (*path-based*) и *гибридные* (*hybrid*).

Алгоритмы на *основе соединений* можно разделить на 2 подгруппы:

- методы, использующие *весовые коэффициенты* (*net-weighting*),

- методы, *накладывающие ограничения на соединения* (*net constraints*).

Ограничивающие методы задают предельное значение длины или задержки соединений, гарантируя их оптимизацию. Для получения оптимального размещения с точки зрения быстродействия

необходимо, чтобы ограничения не приводили к чрезмерному уменьшению пространства возможных решений, вследствие этого ограничивающие методы не настолько точны, как методы, использующие весовые коэффициенты.

Рассмотрим методы, использующие *весовые коэффициенты.* В них информация о задержках, полученная после статического временного анализа, используется для задания весов межсоединениям в ходе размещения. Чем выше вес, тем выше вероятность того, что данное соединение окажется на критическом пути. Веса могут оставаться как *статическими* в ходе размещения (*static net weights*), так и *динамически* меняться (*dynamic net weights*).

Статические методы имеют ряд недостатков, например, уменьшая задержку критических путей, статические методы создают новые критические пути, что в отдельных случаях может привести к ухудшению быстродействия. Подход с изменением весов более гибок, но также не лишен недостатков. Обновление весов может приводить к осцилляциям, когда цепь начинает выходить из группы критических соединений и возвращаться обратно после каждого обновления. Для устранения таких осцилляций может быть введена история весов для каждого соединения, которая будет учитываться при изменении веса.

Другой подход используется в методах на *основе путей* [6]. Данные методы намного точнее методов на основе соединений и выполняют непосредственную оптимизацию задержек на всех критических путях. Задача оптимизации решается как задача поиска кратчайшего пути. Новое размещение для элементов на критическом пути ищется таким образом, чтобы уменьшить суммарную задержку на всем пути без понижения минимального слека. Несмотря на точность методов на основе путей, данный подход к оптимизации быстродействия плохо масштабируется, т.к. количество возможных путей в схеме растет экспоненциально с ростом количества элементов в схеме.

В данной работе был предложен алгоритм размещения с оптимизацией быстродействия на основе метода имитации отжига. Была разработана новая функция оценки, основанная на модели полупериметра охватывающего прямоугольника и новой матричной модели задержек. Применение разработанного алгоритма позволило повысить быстродействие схем на РСнК.

Статья организована следующим образом: в Разделе II подробно рассмотрен предложенный алгоритм размещения, экспериментальные результаты и их анализ представлены в Разделе III, с последующими выводами в Разделе IV.

II. Разработанный алгоритм

В рамках данной работы предложен метод размещения с оптимизацией быстродействия для РСнК «А01» с островной архитектурой, которая подробно описана в работе [6]. Учитывая иерархичность структуры, проводится отдельное размещение для каждого уровня иерархии. Таким образом, выделяются два этапа: глобальный и детальный. Схематичное описание предложенного метода представлено на рис. 1.

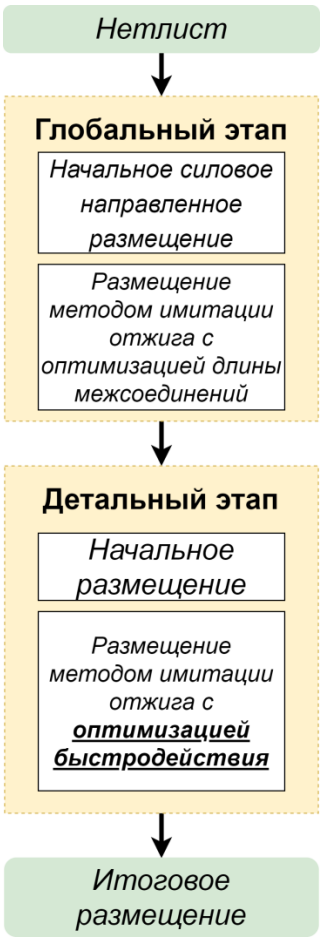

Рис. 1. Схематичное описание разработанного метода

A. Глобальный этап

На глобальном этапе выполняется размещение групп логических элементов с оптимизацией длин цепей и учетом особенностей архитектуры ПЛИС. Группы логических элементов (ГЛЭ) формируются с помощью алгоритма кластеризации, представленного в [7]. Сначала генерируется начальное решение с использованием силового алгоритма размещения с учетом привязки к ячейкам ввода/вывода и макроблокам. Далее проводится оптимизация начального решения.

Для вычисления начальной и конечной температуры и количества итераций используется метод, описанный в [8]. На каждой итерации проводится взаимная перестановка ГЛЭ либо перемещение ГЛЭ на свободную позицию.

Архитектура РСнК «А01» выполнена в островном стиле. Таким образом, конфигурируемые логические блоки разделяются на массивы (ЛАБ) по 256 элементов каждый. Для соединения элементов из разных блоков

используются глобальные трассы. Из-за ограниченного количества использование глобальных трасс должно строго контролироваться. Поэтому для оптимизации количества используемых трассировочных ресурсов была разработана следующая целевая функция:

$$Cost = \sum_{n \in Nets} A(n) \times \rho(n), \quad (3)$$

где $\rho(n)$ – это штрафной коэффициент, а $A(n)$ – это целевая функция выравнивания блоков, которая рассчитывается по следующей формуле (рис. 2):

$$A(n) = (bb_x(n)+1) \times (bb_y(n)+1), \quad (4)$$

где $bb_x(n)$, $bb_y(n)$ – ширина и высота охватывающего прямоугольника цепи n. Целевая функция $A(n)$ позволяет уменьшить задержки за счет уменьшения количества используемых трассировочных элементов.

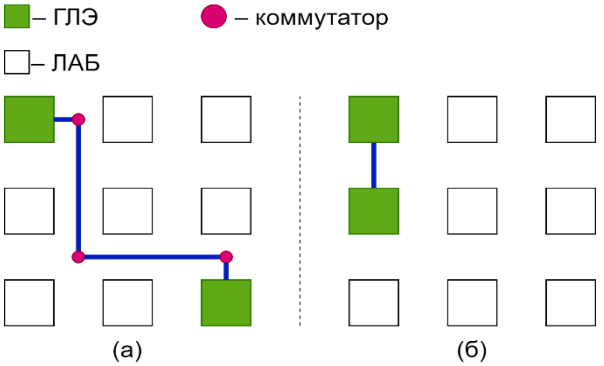

Рис. 2. Значение целевой функции выравнивания блоков для (а) невыравненного и (б) выравненного размещений. (а) $A(n)$ = (2+1)×(2+1)=9. (b) $A(n)$ = (0+1)×(1+1)=2

Штрафной коэффициент $\rho(n)$ предназначен для выравнивания элементов относительно макроблоков и ячеек ввода-вывода (ЯВВ) и вычисляется по следующей формуле:

$$\rho(n) = \sum_{l \in LabPins(n)} t^l_{ij}, \quad (5)$$

где $LabPins(n)$ – это множество ГЛЭ, связанных с цепью n, T – матрица, в которой хранятся значения штрафных коэффициентов, а i и j – индексы в матрице, соответствующие относительным координатам ГЛЭ по вертикали и горизонтали. Элементы матрицы вычисляются по следующей формуле:

$$t^l_{ij} = E(p) - S(p), \quad (6)$$

где $E(p)$ – количество соединений ГЛЭ с макроблоками/ЯВВ и $S(p)$ – количество связанных с ГЛЭ макроблоков/ЯВВ, размещенных в i-строке или j-ом столбце (рис. 3). Если в матрице содержатся отрицательные значения, то выполняется нормализация всех элементов по следующей формуле:

$$t_{ij}^{l,new} = t_{ij}^{l,old} + \left| \min_{\forall i,j} t^l_{ij} \right|. \quad (7)$$

Благодаря штрафному коэффициенту $\rho(n)$ уменьшается количество используемых глобальных трасс, что также оптимизирует задержку схем, т.к. уменьшает количество используемых трассировочных элементов.

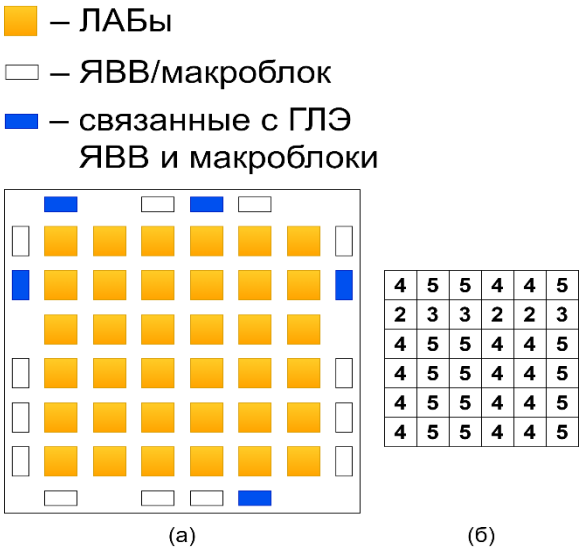

Рис. 3. Матрица Т для заданной ГЛЭ. (а) Размещение связанных с ГЛЭ макроблоков и ячеек ввода-вывода. (б) Матрица Т

B. Детальный этап

Детальное размещение следует после глобального размещения. На данном этапе размещаются логические элементы внутри каждой ГЛЭ. На подготовительной фазе определяется порядок обработки ГЛЭ, ряды обрабатываются в порядке сверху вниз, в пределах ряда ГЛЭ, при этом элементы, расположенные левее, имеют более высокий приоритет.

Когда порядок определен, генерируется начальное размещение, описанное в [6], после чего проводится его оптимизация на основе метода имитации отжига. Алгоритмы на основе метода имитации отжига отлично подходят для оптимизации размещения компонентов РСнК, в силу ограниченного пространства поиска возможных решений и при хорошо подобранных параметрах они способны найти оптимальное размещение за приемлемое время.

В данной статье мы предлагаем новую целевую функцию для оптимизации быстродействия схемы.

$$Cost = \sum_{n=1}^{Nnets} Timing\_cost_n \times q(n) \times \left[bb_x(n) + bb_y(n) \right], \quad (8)$$

где $q(n)$ – коэффициент, нивелирующий тот факт, что модель охватывающего прямоугольника искажает реальную длину цепей для цепей с количеством терминалов больше 3-х [8].

Для вычисления $TimingCost_n$ используются матрицы задержек. Они учитывают задержки распространения сигнала между выводами схемы. Матрицы строятся для

пары источник-приемник с учетом типа соединения. Использование матричной записи обусловлено ее эффективностью для хранения данных и возможностью быстрой их обработки.

Для ПЛИС характерно наличие разных типов коммутационных ресурсов. В ПЛИС «A01» таких типов два: быстрые локальные связи и глобальные шины. Локальные связи используются для соединения элементов, находящихся в пределах одного блока. Глобальные шины используются для соединения логических ячеек, расположенных в разных блоках логических ячеек.

В ряде исследований было установлено, что величина задержки в ПЛИС зависит от количества используемых сегментов и коммутаторов в цепи [9]. Поэтому в матрицу задержек вносятся не сами значения задержек, а количество сегментов/коммутаторов (рис. 4). Реальную задержку при необходимости можно будет восстановить из функции зависимости задержки от количества коммутаторов.

Сегментированная структура локальных связей позволяет использовать минимально возможное количество отдельных сегментов, что положительно отражается на быстродействии.

	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8
-7	10	10	10	9	8	8	6	6	6	8	8	8	8	8	9	9
-6	10	9	9	9	7	7	6	6	6	7	7	7	8	8	8	8
-5	10	9	8	8	7	6	6	6	6	6	6	6	6	6	8	8
-4	9	8	8	7	6	6	4	4	4	6	6	6	6	6	7	7
-3	9	7	7	7	5	5	4	4	4	5	5	5	6	6	6	6
-2	8	7	6	6	5	4	4	4	4	4	4	4	4	4	6	6
-1	7	6	6	5	4	4	2	2	2	4	4	4	4	4	6	6
0	7	6	6	5	4	4	2	0	2	2	2	2	4	4	4	6
1	6	6	6	4	4	4	2	2	2	4	4	4	5	5	6	7
2	8	6	6	6	4	4	4	2	4	4	5	5	5	5	7	7
3	8	7	6	6	5	4	4	2	4	4	5	5	6	7	7	8
4	8	7	6	6	5	4	4	2	4	4	5	6	6	7	8	8
5	8	8	6	6	6	4	4	4	4	4	6	6	6	7	8	8
6	8	8	6	6	6	4	4	4	4	4	6	6	6	8	8	8
7	8	8	8	6	6	6	6	4	6	6	6	7	8	8	9	10
8	9	8	8	7	7	6	6	6	6	6	7	7	8	9	9	10

Рис. 4. Матрица задержек

Структура межсоединений ПЛИС может быть несимметричной, например в ПЛИС «A01» наблюдается правосторонность локальных связей, что учтено в используемой матрице. Размерность матрицы задается формулой:

$$\begin{cases} n_{rows} = 2 \times LE_{row} - 1, \\ n_{columns} = 2 \times LE_{colums} - 1, \end{cases} \qquad (9)$$

где LE_{rows} и $LE_{columns}$ - это количество ЛЭ, объединенных в один блок по строкам и столбцам, соответственно. Источник сигнала закреплен в центре матрицы и имеет координаты (i_{src}, j_{src}). Приемник располагается на расстоянии $(\Delta i, \Delta j)$ от источника, если это расстояние не превышает размер блока логических элементов. Его координаты определяются как $(i_{src}+\Delta i, j_{src}+\Delta j)$. Для всех возможных значений $(\Delta i, \Delta j)$ проводится трассировка [10], а полученное число коммутаторов заносится в матрицу.

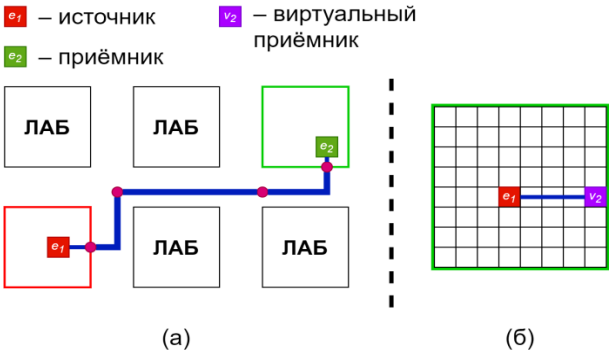

Рис. 5. Создание виртуального элемента

Если источник и приёмник расположены в разных ГЛЭ, то использование глобальных шин неизбежно. При этом сохраняется возможность уменьшить долю задержки, приходящуюся на локальную связь. Для этого вводится понятие виртуального источника/приемника, который должен соединяться с глобальной шиной минимальным возможным количеством коммутаторов и располагаться в той же ГЛЭ, где выполняется детальное размещение. Задержка соединения приемника/источника с виртуальным приемником будет определяться матрицей задержек для соединений внутри ГЛЭ. Расположение виртуального элемента зависит от его размещения в ГЛЭ и размещения ГЛЭ друг относительно друга. Описанный принцип действия проиллюстрирован на рис. 5.

III. Результаты

Предложенный алгоритм был реализован на языке C с использованием компилятора gcc 4.1. Для тестов использовалась машина под управлением RHEL 5 с процессором Intel Xeon X5650 (2,67 ГГц, 6 ядер) и оперативной памятью емкостью 8 Гб.

Для оценки эффективности предлагаемый алгоритм был протестирован на наборах тестовых схем ISCAS-85/89 [11]. Для сравнения использовался алгоритм с функцией оценки, оптимизирующей длину межсоединений, Star+ [12]. Размещение на глобальном этапе производилось одним и тем же алгоритмом, описанным в разделе IIa этой статьи.

В табл. 1 представлены результаты размещения с оптимизацией быстродействия и с оптимизацией длины межсоединений. Полученные результаты показывают, что в большинстве тестов разработанный алгоритм улучшает быстродействие тестовых схем. В наилучшем случае улучшение составляет 32% для быстродействия и 10% - для длины межсоединений.

Таблица 1

Сравнение задержки критического пути и суммарной длины межсоединений для версии с оптимизацией длины межсоединений и версии с оптимизацией быстродействия

Схема	***ЛЯи / ЛАБы***	***Алгоритм***	***Длина межсоединений, у.е.***	***Δ_w,%***	***Задержка, нс***	***Δ_d,%***
s641	97/1	*Star+*	2231	-4	107	30
		Разработанный	2337		82	
s820	163/2	*Star+*	1644	-13	70	17
		Разработанный	1902		59	
c1355	259/3	*Star+*	4718	0.4	102	13
		Разработанный	4699		90	
s1488	346/6	*Star+*	4508	-8	123	32
		Разработанный	4902		93	
c3540	540/7	*Star+*	10115	10	183	27
		Разработанный	9164		143	
s5378	756/13	*Star+*	13574	-15	104	-8
		Разработанный	16133		114	
c6288	1145/18	*Star+*	10041	-36	401	23
		Разработанный	15911		325	
s38417	5543/87	*Star+*	97099	-25	295	10
		Разработанный	130743		266	

IV. Заключение

В данной статье предлагается новый алгоритм многоуровневого размещения с оптимизацией быстродействия для РСнК с архитектурой островного типа, основанный на модели матриц задержек. Размещение проводится в 2 этапа: на глобальном и детальном уровне. На этапе глобального размещения используется силовое направленное размещение с последующей оптимизацией на основе метода имитации отжига. Далее проводится детальное размещение с оптимизацией быстродействия, также основанное на методе имитации отжига. Разработанная оценочная функция позволяет сократить задержки в схеме за счет уменьшения количества коммутаторов в каждой трассируемой цепи, для этого используется заранее подготовленная матрица задержек.

Экспериментальное сравнение с версией, оптимизирующей длину межсоединений, приведено в табл. 1. Разработанный алгоритм доказал свою эффективность, уменьшив задержки в среднем на 18%, при повышении длины межсоединений в среднем на 11%.

Литература

[1] Hauck S., DeHon A. Reconfigurable Computing: The Theory and Practice of Fpga-Based Computation San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., 2007.

[2] Hitchcock R., Smith G., Cheng D. Timing Analysis of Computer-Hardware // IBM Journal of Research and Development. Jan. 1983, pp. 100-105. DOI: 10.1147/rd.261.0100

[3] Гаврилов С.В., Иванова Г.А. Анализ быстродействия сложных цифровых схем с учетом неопределенности технологических и схемных параметров // Вестник Рязанского государственного радиотехнического университета. 2015. Вып. 53. С. 29-35.

[4] Markov I.L., Hu J., Kim M. Progress and Challenges in VLSI Placement Research // Proceedings of the IEEE, Nov. 2015, vol. 103, no. 11, pp. 1985-2003. DOI: 10.1109/JPROC.2015.2478963

[5] Лебедев Б. К., Степаненко С. А. Генетический алгоритм размещения, управляемый временными ограничениями // Известия ЮФУ. Технические науки. 2007. №1. URL: https://cyberleninka.ru/article/n/geneticheskiy-algoritm-razmescheniya-upravlyaemyy-vremennymi-ogranicheniyami (дата обращения: 08.04.2020).

[6] Gavrilov S., Zheleznikov D., Chochaev R. Simulated Annealing Based Placement Optimization for Reconfigurable Systems-on-Chip // 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). Saint Petersburg and Moscow, Russia, 2019, pp. 1597-1600. DOI: 10.1109/EIConRus.2019.8657251

[7] 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. Вып. 1. С. 199-204. DOI: 10.31114/2078-7707-2018-1-199-204

[8] Betz V., Rose J. VPR: A new packing, placement and routing tool for FPGA research // Proceedings of the 7th International Workshop on Field-Programmable Logic and Applications, ser. FPL '97. 1997, pp. 213-222. DOI :10.1007/3-540-63465-7_226

[9] Chang Y.-W., Zhu K., Wong D. F. Timing-driven routing for symmetrical-array-based FPGAs // ACM TODAES. July 2000, vol. 5, no. 3, pp. 433-450. DOI: 10.1109/ICCD.1998.727132

[10] Гаврилов С.В., Железников Д.А., Заплетина М.А., Хватов В.М., Чочаев Р.Ж., Эннс В.И. Маршрут топологического синтеза для реконфигурируемых систем на кристалле специального назначения //

Микроэлектроника. 2019. Т. 48. № 3. С. 211-223. DOI: 10.1134/S0544126919030050
[11] Bryan D. The ISCAS ‘85 Benchmark Circuits and Netlist Format. North-Carolina State University, 1985.
[12] Xu M., Grewal G., Areibi S. StarPlace: A new analytic method for FPGA placement // Integr. VLSI J. June 2011. 44 (3), pp. 192-204. DOI: 10.1016/j.vlsi.2011.02.001

Timing-Driven Placement Algorithm Based on Delay Matrix Model for Reconfigurable System-on-Chip

P.I. Frolova, R.Zh. Chochaev, G.A. Ivanova, S.V. Gavrilov

Institute for Design Problems in Microelectronics of RAS, Moscow,

frolova_p@ippm.ru, chochaev_r@ippm.ru, ivanova_g@ippm.ru, s.g@ippm.ru

***Abstract* — Placement is one of the most difficult stages of reconfigurable system-on-chip design flow. Designing high-speed systems requires efficient timing-driven placement algorithms.**

In this article we present a new timing-driven placement algorithm based on simulated annealing method for the island-style RSoC. Since the island-style RSoC is hierarchical, our algorithm is divided in two stages: global and detailed. At the global stage we place groups of logic elements (GLE) with respect to the assigned input/output cells and macroblocks. At the detailed stage we place logic elements inside each GLE. We developed new cost function for detailed placement that takes into account number of switches.

To minimize critical path delay we use delay lookup matrices. We use the number of switches in the routed path to predict pin-to-pin interconnection delay. To calculate the number of switches, we place two elements (source and sink) with a distance (Δi, Δj) and route them with the same router that will be used at the final routing.

To route two elements placed in different GLEs, we define virtual element in the path. Virtual sink is placed in the same GLE with source. We calculate amount of switches between source and virtual sink and add extra fine for global bus. Virtual element placement ensures shortest path from source to sink.

Experimental results showed timing improvement by an average 18% and increased wirelength by an average 11% compared to wirelength-driven algorithm based on Star+ model.

Proposed algorithm showed its efficiency in high-speed RSoC design flow.

***Keywords* — FPGA, reconfigurable system-on-chip (RSoC), electronic design automation (EDA).**

REFERENCES

[1] Hauck S., DeHon A. *Reconfigurable Computing: The Theory and Practice of Fpga-Based Computation* San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., 2007.
[2] Hitchcock R., Smith G., Cheng D. Timing Analysis of Computer-Hardware // *IBM Journal of Research and Development.* Jan. 1983, pp. 100-105. DOI: 10.1147/rd.261.0100
[3] Gavrilov S.V., Ivanova G.A. Analiz bystrodejstvija slozhnyh cifrovyh shem s uchetom neopredelennosti tehnologicheskih i shemnyh parametrov (Timing analysis of complex digital circuits with the process uncertainty and circuit parameters) // Vestnik Rjazanskogo gosudarstvennogo radiotehnicheskogo universiteta. 2015. (№ 53), pp. 29-35.
[4] Markov I.L., Hu J., Kim M. Progress and Challenges in VLSI Placement Research // *Proceedings of the IEEE*, Nov. 2015, vol. 103, no. 11, pp. 1985-2003. DOI: 10.1109/JPROC.2015.2478963
[5] Lebedev B. K., Stepanenko S. A. Geneticheskij algoritm razmeshcheniya, upravlyaemyj vremennymi ogranicheniyami (Timing-driven genetic placement algorithm) // Izvestiya YUFU. Tekhnicheskie nauki. 2007. №1. URL: https://cyberleninka.ru/article/n/geneticheskiy-algoritm-razmescheniya-upravlyaemyy-vremennymi-ogranicheniyami (data obrashcheniya: 08.04.2020) (in Rassian).
[6] Gavrilov S., Zheleznikov D., Chochaev R. Simulated Annealing Based Placement Optimization for Reconfigurable Systems-on-Chip // *2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, Saint Petersburg and Moscow, Russia, 2019, pp. 1597-1600. DOI: 10.1109/EIConRus.2019.8657251
[7] Gavrilov S.V., Zheleznikov D.A., Chochaev R., Khvatov V.M. Partitioning Algorithm Based on Simulated Annealing for Reconfigurable Systems-on-Chip // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2018. Issue 1. P. 199-204. DOI: 10.31114/2078-7707-2018-1-199-204 (in Rassian).
[8] Betz V., Rose J. VPR: A new packing, placement and routing tool for FPGA research // *Proceedings of the 7th International Workshop on Field-Programmable Logic and Applications, ser. FPL '97*, 1997, pp. 213-222. DOI :10.1007/3-540-63465-7_226
[9] Chang Y.-W., Zhu K., Wong D. F. Timing-driven routing for symmetrical-array-based FPGAs // *ACM TODAES*, July 2000, vol. 5, no. 3, pp. 433-450. DOI: 10.1109/ICCD.1998.727132
[10] Gavrilov S.V., Zheleznikov D.A., Zapletina M.A., Chochaev R. Z., Enns V.I. Layout Synthesis Design Flow for Special-Purpose Reconfigurable Systems-on-a-Chip // *Russian Microelectronics*, 2019, vol. 48, is.3, pp. 176-186. DOI:10.1134/s1063739719030053 (in Rassian).
[11] Bryan D. The ISCAS ‘85 Benchmark Circuits and Netlist Format. North-Carolina State University, 1985.
[12] Xu M., Grewal G., Areibi S. StarPlace: A new analytic method for FPGA placement // Integr. VLSI J. June 2011. 44 (3), pp. 192-204. DOI: 10.1016/j.vlsi.2011.02.001.