УДК 621.3.049.771 DOI: 10.31114/2078-7707-2020-4-39-44

Метод логического ресинтеза схем в маршруте проектирования на ПЛИС

Н.О. Васильев, И.В. Тиунов, Д.И. Рыжова

Институт проблем проектирования в микроэлектронике РАН, г. Москва, г. Зеленоград

vasilyev_n@ippm.ru, tiunov_i@ippm.ru, ryzhova_d@ippm.ru

***Аннотация* — В работе предлагается метод логического ресинтеза технологических отображений на ПЛИС в процессе преобразования исходного описания логической схемы в описание на базе структурных элементов ПЛИС. При этом необходимо, чтобы полученное описание было оптимизированно по различным параметрам. Метод работает над графовой моделью, представляющей исходную логическую схему, над узлами которой определены логические операции для изменения структуры графа. Существующие подходы зачастую используют жадные алгоритмы для быстрого получения результата. Однако это может привести к тому, что полученный результат будет являться локальным минимумом целевой функции. В основе предлагаемого метода лежит алгоритм моделирования отжига, предполагающий возможность принятия плохих решений для избегания локальных минимумом целевой оценочной функции. Тестирование разработанного метода показало уменьшение количества требуемых для отображения логических элементов ПЛИС в среднем на 15% по сравнению с начальным отображением, полученным с помощью алгоритма FlowMap.**

***Ключевые слова* — ПЛИС (Программируемые логические интегральные схемы), LUT, технологическое отображение, ресинтез.**

I. Введение

Одним из факторов популярности ПЛИС является возможность гибкой настройки путем перепрограммирования. Это возможно благодаря использованию основного структурного компонента ПЛИС – программируемых таблиц поиска (анг. Look-Up Table, LUT) в качестве базового логического элемента. LUT содержит 2^k конфигурационных бит, которые могут быть использованы для реализации любой Булевой функции от k-переменных.

В маршруте проектирования на ПЛИС одним из важнейших этапов является этап технологического отображения технологически независимого HDL описания в технологически зависимое на основе LUT. В ходе данного процесса также производится оптимизация структуры схемы по таким параметрам, как длина критического пути, занимаемая на ПЛИС площадь и количество межсоединений.

Для оптимизации решения по необходимым критериям существуют различные подходы, которые можно разделить на две группы: зависящие и независящие от внутренней архитектуры конкретной ПЛИС. К первой группе относится метод объединения LUT и триггеров в единый программируемый логический блок [1]. Подобные оптимизации совершаются с учетом внутренней архитектуры конкретного семейства ПЛИС. К второй группе относится широкий набор методов логической оптимизации схем. Популярным методом, относящимся к данной группе, является поиск оптимальных разрезов на графе [2]. При использовании подобных методов необходимо учитывать широкий набор возможных вариантов декомпозиций логических элементов. Кроме того, подобные методы используют упрощенные модели для ускорения вычислений.

В данной статье рассматривается алгоритм архитектурно-независимого ресинтеза логических схем, прошедших этап предварительного отображения на ПЛИС. Ресинтез представляет из себя процесс оптимизации схем, синтезированных стандартными методами, для улучшения различных параметров [3] или модификации синтезированного описания после изменения спецификаций схемы [4]. Предложенный алгоритм ресинтеза основан на алгоритме структурной оптимизации для микроэлектронных схем на транзисторном уровне [3, 5].

II. Графовая модель

Логические схемы в предлагаемом методе представляются в виде направленного ациклического графа $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$, узлы $\boldsymbol{V}$ которого соответствуют логическим элементам схемы, а направленные ребра $\boldsymbol{E}$ соответствуют соединениям между этими элементами. Каждый узел $v \in \boldsymbol{V}$ хранит Булеву функцию соответствующего логического элемента (рис. 1). Обозначим с помощью $\boldsymbol{I_v} \in \boldsymbol{E}$ входное множество ребер узла $v \in \boldsymbol{V}$, а с помощью $\boldsymbol{O_v} \in \boldsymbol{E}$ – выходное множество ребер этого узла. При этом, $O_{v_i} \in \boldsymbol{O_v}$ – обозначает i-ый элемент выходного множества.

Для каждого узла графа $v \in \boldsymbol{V}$ определяется параметр «уровень узла» ($level(v)$). Уровень узла – это максимальное количество узлов на самом длинном пути от первичных входов до этого узла. Максимальный уровень узла в графе определяет глубину этого графа ($depth(\boldsymbol{G})$). Благодаря глубине графа можно дать грубую, но в то же время быструю оценку критического пути [6].

Так как ребро графа соответствует соединению между двумя элементами, то количество ребер графа определяет количество межсоединений. Уменьшение длины межсоединений позволяет улучшить трассируемость схемы.

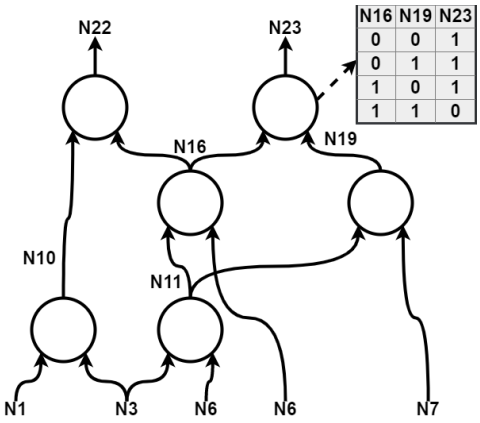

Рис. 1. Граф для схемы c17 из набора ISCAS-85

Над узлами графа определены следующие логические операции:

- декомпозиция узлов;
- слияние узлов;
- переупорядочивание узлов.

Данные операции изменяют логическую структуру графа, сохраняя при этом неизменной Булеву функцию схемы на первичных выходах. Рассмотрим подробнее каждую из операций.

A. Декомпозиция

Декомпозиция (или разложение) – логическая операция для представления логической функции ***f(X)*** в виде набора меньших подфункций [7]. Данная операция может быть записана в следующем виде:

$$f(\boldsymbol{X}) = h(g_1(\boldsymbol{X_1}), g_2(\boldsymbol{X_1}), \dots, g_k(\boldsymbol{X_1}), \boldsymbol{X_2}),$$

где $\boldsymbol{X} = \boldsymbol{X_1} \cup \boldsymbol{X_2}$, $k < |\boldsymbol{X_1}|$.

Если $\boldsymbol{X_1} \cap \boldsymbol{X_2} = \emptyset$, то такая операция называется декомпозицией на непересекающиеся множества (рис. 2); иначе – декомпозицией на пересекающиеся множества (рис. 3). Если k = 1, то декомпозиция называется простой. Множество $\boldsymbol{X_1}$ называется связанным, множество $\boldsymbol{X_2}$ свободным. Функция g называется предком, функция h – потомком.

Одним из способов поиска возможных вариантов разложений логической функции является поиск по таблице разложения [8]. Таблица разложения похожа на карту Карно, но не использует код Грея для индексов рядов и колон.

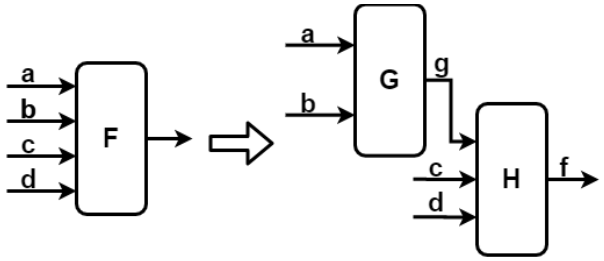

Рис. 2. Декомпозиция на непересекающиеся множества

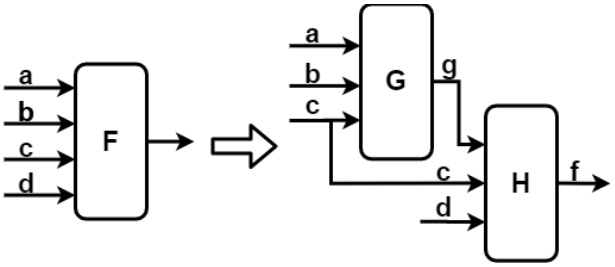

Рис. 3. Декомпозиция на пересекающиеся множества

Две ячейки таблицы разложения называются совместимыми, если они имеют одинаковые значения, либо значение хотя бы одной из них неопределенно. Две колонки называются совместимыми, если все ячейки в них, индексы ряда которых совпадают, совместимы. Количество несовместимых колонок (рядов) таблицы разложения называется сложностью колонок (рядов) и обозначается $\mu(X_1|X_2)$. На рис. 4. показана таблица разложения для следующей Булевой функции:

$$f(a, b, c, d) = a\bar{b} + \bar{b}\bar{c} + \bar{b}d + \bar{a}bc\bar{d}.$$

Сложность колонок (рядов) позволяет определить с помощью теоремы Кёртиса возможно ли разложить Булеву функцию на заданные свободное и связанное множества [9]. Согласно теореме Кёртиса декомпозиция на заданные множества возможна только в том случае, если сложность колонок не превышает 2^k, где k – количество функций-предков.

ab \ cd	00	01	10	11
00	1	1	0	1
01	0	0	1	0
10	1	1	1	1
11	0	0	0	0
	A	A	B	A

Рис. 4. Таблица разложения

Свободное множество для данной таблицы – $\boldsymbol{X_1} = \{C, D\}$, связанное множество – $\boldsymbol{X_2} = \{A, B\}$. Так как столбцы 00, 01, 11 совместимы между собой, а столбец 10 несовместим с ними, то сложность данной таблицы

разложения равна 2. Значит для заданного свободного множества достаточно одной функции-предка. С помощью данной таблицы можно найти логическую функцию для свободного множества (рис. 5.), а с помощью нее – функцию для связанного множества.

ab \ cd	00	01	10	11	
00	1	1	0	1	g_1
01	0	0	1	0	$\overline{g_1}$
10	1	1	1	1	1
11	0	0	0	0	0

Рис. 5. Функция для свободного множества

Таким образом, функция-предок будет выглядеть следующим образом:

$$g(c,d) = \bar{c}|d.$$

Функция-потомок имеет вид:

$$h(a,b,c) = \bar{a}\bar{b}g|\bar{a}b\bar{g}|a\bar{b}.$$

B. Слияние

Слияние узлов графа [10] является обратной операцией для простого разложения. Эта операция комбинирует два связанных узла в один.

Определим два узла графа $A, B \in \boldsymbol{V}$ такие, что выход узла A принадлежит входному множеству узла B, т.е. $O_{A_i} \in \boldsymbol{I_B}$. Тогда слияние может быть записано следующим образом:

$$h(g(\boldsymbol{I_A}), \boldsymbol{I_B}) = f(\boldsymbol{I_A} \cup \boldsymbol{I_B}),$$

где g и h – логические функции узлов A и B соответственно. При этом, если выход узла A также принадлежит входному множеству других элементов, т.е. $O_{A_j} \in \boldsymbol{I_C}$, то перед слиянием узлов A и B необходимо произвести дублирование узла A.

C. Переупорядочивание

Операция переупорядочивания меняет местами два узла графа. Переупорядочивание также, как и слияние производится над двумя узлами графа $A, B \in \boldsymbol{V}$, однако выходное ребро узла A должно принадлежать только входному множеству узла B, т.е. $O_{A_i} \in \boldsymbol{I_B}, |\boldsymbol{O_A}| = 1$.

Данная операция позволяет изменять уровень узлов не меняя количество узлов, что ведет к изменению глубины графа. Кроме того, изменение порядка узлов может привести к появлению новых вариантов для остальных логических операций.

Фактически, данная операция производит сначала слияние двух соединенных узлов, а затем их декомпозицию, но поменяв свободное и связанное множества местами.

III. Ресинтез

В предложенном алгоритме ресинтеза можно выделить два главных цикла: глобальный и локальный.

Локальный алгоритм ресинтеза работает над небольшими фрагментами исходного графа, которые называются окнами $\boldsymbol{W(V,E)}$ [11]. Выбор окна производится исходя из необходимых потребностей.

Для минимизации глубины графа, а вследствие этого и длины критического пути, в окно берутся все узлы, которые находятся на этом пути. Для минимизации площади или количества межсоединений выбор окна производится в топологическом порядке от первичных входов к первичным выходам. В этом случае существует ограничение на размер окна, которое определяется количеством листьев в выбранном подграфе. При этом избегается критический путь.

Локальный алгоритм основан на вероятностном алгоритме моделирования отжига. Данный алгоритм допускает принятие таких решений, которые ведут к ухудшению целевой оценочной функции. Однако это необходимо для того, чтобы избегать ее локальные минимумы. Алгоритм отжига доказал свою эффективность на различных этапах маршрута проектирования на ПЛИС [12, 13].

Для оценки получаемых в ходе ресинтеза решений используется целевая функция следующего вида:

$$Cost = K * \frac{area}{area_0} + (1-K)\frac{edges}{edges_0} + \beta \cdot \Delta depth,$$

где $area$ – текущее количество узлов графа, $area_0$ – начальное количество узлов графа, $edges$ – текущее количество ребер графа, $edges_0$ – начальное количество ребер графа, K – весовой коэффициент, задающий направление улучшения, $\beta \cdot \Delta depth$ - штраф, накладываемый на целевую функцию в случае увеличения глубины графа.

На каждой итерации выбирается произвольный узел внутри локального окна. Над выбранным узлом производится одна из логических операций, описанных ранее. Если новое решение ведет к улучшению целевой функции, то оно принимается. Если стоимость целевой функции ухудшится вследствие принятия нового решения, то рассчитывается вероятность принятия плохого решения. Данная вероятность зависит от температуры отжига, которая уменьшается с каждой итерацией. Вероятность достигает своего максимума и близка к 1 вначале отжига и постепенно уменьшается. При температурах отжига близких к нулю принимаются только те решения, которые ведут к улучшению стоимости целевой функции.

Во время локального ресинтеза для каждой из логических операций существуют приоритеты. Для декомпозиции приоритеты операций выглядят следующим образом:

- простая декомпозиция на непересекающиеся множества;
- простая декомпозиция на пересекающиеся множества;
- сложная декомпозиция.

Для операции слияния назначены следующие приоритеты:

- слияние без дублирования узла;
- слияние с дублированием узла.

Приоритеты операций выбраны исходя из того, какое влияние они оказывают на количество узлов в графе и количество использованных входов в каждом из образующихся в ходе операции узлов.

Локальный ресинтез может быть представлен в виде следующего алгоритма:

$T = T_{max}$
Пока $T > T_0$
  Выбрать вершину $v_W \in \boldsymbol{V_W}$
  Определить возможные операции над v_W
  Произвести одну из логических операций над v_W
  Получить новую оценку целевой функции *newCost*
  Если $(newCost < bestCost)$
    Принять изменение
  Иначе если $P(\Delta Cost, T) > P_0)$
    Принять изменение
  Иначе
    Отклонить изменение
  Уменьшить T

Глобальный ресинтез работает над всем графом сразу. Это позволяет при выполнении каждой из операций учитывать состояние остальных узлов графа. Так, например, при выполнении операции декомпозиции образуются новые узлы. Однако нередко возникают ситуации, когда в графе присутствуют узлы с таким же входным множеством и аналогичной Булевой функцией (с учетом неопределенных состояний). Данные узлы могут быть переиспользованы. Благодаря этому декомпозиция не увеличит число узлов графа. Данный алгоритм выглядит следующим образом:

Покрыть $\boldsymbol{G(V, E)}$ окнами $\boldsymbol{W(V_w, E_w)}$
$\forall \boldsymbol{W_0} \in \boldsymbol{W(V_W, E_W)}$
  Произвести локальный ресинтез для $\boldsymbol{W_0}$
  Если $newCost < bestCost$
    Заменить исходный подграф полученным
    Найти и удалить эквивалентные вершины

IV. Результаты тестирования

Разработанный метод был протестирован на наборе тестовых схем ISCAS-85. Данные схемы были предварительно отображены с помощью алгоритма FlowMap [14], предназначенного для минимизации глубины схемы. Направлением улучшения было выбрано уменьшение количества узлов без увеличения глубины графа. Результаты представлены в таблицах 1, 2 для 3-входовых и 4-входовых LUT (3-LUT и 4-LUT), соответственно.

Таблица 1

Результаты для 3-LUT

Схема	area/startArea	Улучшение, %
C17	0,80	20%
C432	0,81	19%
C499	0,90	10%
C880	0,76	24%
C1355	0,85	15%
C1908	0,97	3%
C2670	0,77	23%
C3540	0,87	13%
C5315	0,71	29%
C6288	0,82	18%
C7552	0,96	4%
Average value	**0,84**	**16%**

Таблица 2

Результаты для 4-LUT

Circuit	area/startArea	Profit, %
C17	1,00	0%
C432	0,73	27%
C499	1,00	0%
C880	0,69	31%
C1355	1,00	0%
C1908	0,99	1%
C2670	0,76	24%
C3540	0,78	22%
C5315	0,70	30%
C6288	0,82	18%
C7552	0,94	6%
Average value	**0,85**	**15%**

Также была проверена зависимость результатов ресинтеза от количества итераций глобального алгоритма. Результаты представлены в таблице 3.

Можно заметить, что в большинстве случаев наиболее эффективны только первые две итерации. Последующие итерации введут к незначительным улучшениям. Таким образом, можно сделать вывод, что нецелесообразно производить более двух итераций глобального алгоритма.

Таблица 3

Зависимость уменьшения площади от количества итераций

Схема	Итерация				
	1	2	3	4	5
	Улучшение площади, %				
C17	0%	0%	0%	0%	0%
C432	27%	29%	30%	30%	30%
C499	0%	0%	0%	0%	0%
C880	31%	33%	33%	33%	33%
C1355	0%	0%	0%	0%	0%
C1908	1%	2%	2%	2%	2%
C2670	24%	28%	29%	29%	29%
C3540	22%	25%	25%	25%	25%
C5315	30%	34%	34%	34%	34%
C6288	18%	20%	20%	21%	21%
C7552	6%	7%	8%	8%	8%
Average value	15%	16%	16%	17%	17%

V. Заключение

В статье был предложен метод логического ресинтеза на этапе технологического отображения на ПЛИС. Метод основан на алгоритме моделирования отжига и работает с направленным графом, представляющим исходную логическую схему.

Были получены результаты, демонстрирующие среднее уменьшение количества логических элементов на 16% для 3-LUT и на 15% для 4-LUT. Максимальное улучшение составило 31% по сравнению с исходным отображением.

Также в результате тестирования было выявлено оптимальное количество итераций глобального алгоритма. Исходя из полученных результатов производить более двух итераций не рационально.

Литература

[1] I.V. Tiunov, I.A. Lipatov, D.A. Zheleznikov, “Digital Circuits Resynthesis Approach for FPGAs Based on Logic Cell with Built-In Flip-Flop”, Selected Articles of MES conference, 2018, pp. 33-36. DOI: 10.31114/2078-7707-2019-3-33-36».

[2] A. Mishchenko, S. Cho, S. Chatterjee, R. Brayton, “Combinational and sequential mapping with priority cuts”, ICCAD '07 Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design, 2007, pp. 354-361. DOI: 10.1109/ICCAD.2007.4397290.

[3] S. Gavrilov, D. Ryzhova, N. Vasilyev, T. Zhukova, “Algorithm of Structural Optimization for Digital CMOS Circuits”, Problems of advanced micro- and nanoelectronic systems development, 2019, pp. 42-46. DOI: 10.31114/2078-7707-2019-1-42-46.

[4] A. Stempkovskiy, D. Telpukhov, R. Soloviev, “Fast and accurate resource-aware functional ECO patch generation tool”, 2018 Moscow Workshop on Electronic and Networking Technologies (MWENT 2018), 2018, pp. 1-6. DOI: 10.1109/MWENT.2018.8337192.

[5] S. Gavrilov, G. Ivanova, “Simultaneous Logic and Layout Synthesis for Fin-fet Based Elements with Regular Layout in Polysilicon and Diffusion”, Proceedings of IEEE East-West Design & Test Symposium (EWDTS’2015), 2015, pp. 264-267.

[6] A. Ling, D.P. Singh, S.D. Brown, “FPGA technology mapping: a study of optimality”, DAC '05 Proceedings of the 42nd annual Design Automation Conference, 2005, pp. 427-432. DOI: 10.1145/1065579.1065693.

[7] Y. Li, E. Tsai, M. Perkowski, X. Song, “Grover-based ashenhurst-curtis decomposition using quantum language quipper”, Quantum Information & Computation archive Volume 19 Issue 1-2, February 2019, pp. 35-66.

[8] M. A. Perkowski and S. Grygiel, “A survey of literature on function decomposition,” Portland State University, Tech. Rep. Ver. IV, 1995, p. 188.

[9] H.A. Curtis, “Generalized Tree Circuit - The Basic Building Block of an Extended Decomposition Theory", J. Assn. Comput. Mach, 1963, pp. 562-581.

[10] S.V. Gavrilov, A.L. Glebov, “BDD-based circuit level structural optimization for digital CMOS”, 1-st Intern. Workshop «Multi-Architecture Low Power Design», 1999, P. 45–49.

[11] S.V. Gavrilov, D.I. Ryzhova, N.O. Vasilyev, “Models and methods of inter-gate resynthesis at the transistor level for nanoelectronic circuits based on FinFETs”, 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, 2018, pp. 1364-1367. DOI: 10.1109/EIConRus.2018.8317350.

[12] S. Gavrilov, D. Zheleznikov, V. Khvatov, R. Chochaev, “Clustering Optimization Based on Simulated Annealing Algorithm for Reconfigurable SystemsOn-Chip”, 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2018, pp. 1492-1495. DOI: 10.1109/EIConRus.2018.8317380.

[13] M. Nachtigall, P. Ferreira, F. Marques, “Simulated Annealing applied to LUT-based FPGA Technology Mapping”, 2017 Sixteenth Mexican International Conference on Artificial Intelligence (MICAI), 2017, pp. 23-29. DOI: 10.1109/MICAI-2017.2017.00012.

[14] J. Cong and Yuzheng Ding, "FlowMap: an optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 13, no. 1, 1994, pp. 1-12,. DOI: 10.1109/43.273754. J. Cong and Yuzheng Ding, "FlowMap: an optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 13, no. 1, 1994, pp. 1-12,. DOI: 10.1109/43.27375.

Logic Resynthesis Method in the FPGA Design Flow

N.O. Vasilyev, I.V. Tiunov, D.I. Ryzhova

Institute for Design Problems in Microelectronic of Russian Academy of Sciences (IPPM RAS),

Moscow, Zelenograd

vasilyev_n@ippm.ru, tiunov_i@ippm.ru, ryzhova_d@ippm.ru

***Abstract* — technology mapping is the important stage in the FPGA design flow. The solution obtained at this stage should be optimized for a wide range of criteria. Existing approaches often use greedy algorithms to quickly get results. However, this may lead to the result being a local minimum of the objective function.**

This paper proposes a logical resynthesis method for circuits already mapped on the FPGA. The method works on a graph model, which represent the original logical circuit.

The resynthesis algorithm consists of two cycles: local and global. The local resynthesis algorithm is based on simulated annealing method (SA) and performs optimizations on the subgraphs of the original graph. At each iteration, the algorithm selects one node of the subgraph and performs one of the transformations that are defined on this graph. After this algorithm decides whether to accept the decision. The probability of bad decision acceptance depends on the temperature and on the difference of the cost between the current and new states. Global resynthesis is greedy and works on the whole graph. This allows some optimizations that are not available in the local resynthesis cycle.

***Keywords* — FPGA (Field-Programmable Gate Array), LUT, technology mapping, resynthesis.**

REFERENCES

[1] I.V. Tiunov, I.A. Lipatov, D.A. Zheleznikov, “Digital Circuits Resynthesis Approach for FPGAs Based on Logic Cell with Built-In Flip-Flop”, Selected Articles of MES conference, 2018, pp. 33-36. DOI: 10.31114/2078-7707-2019-3-33-36».

[2] A. Mishchenko, S. Cho, S. Chatterjee, R. Brayton, “Combinational and sequential mapping with priority cuts”, ICCAD '07 Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design, 2007, pp. 354-361. DOI: 10.1109/ICCAD.2007.4397290.

[3] S. Gavrilov, D. Ryzhova, N. Vasilyev, T. Zhukova, “Algorithm of Structural Optimization for Digital CMOS Circuits”, Problems of advanced micro- and nanoelectronic systems development, 2019, pp. 42-46. DOI: 10.31114/2078-7707-2019-1-42-46.

[4] A. Stempkovskiy, D. Telpukhov, R. Soloviev, “Fast and accurate resource-aware functional ECO patch generation tool”, 2018 Moscow Workshop on Electronic and Networking Technologies (MWENT 2018), 2018, pp. 1-6. DOI: 10.1109/MWENT.2018.8337192.

[5] S. Gavrilov, G. Ivanova, “Simultaneous Logic and Layout Synthesis for Fin-fet Based Elements with Regular Layout in Polysilicon and Diffusion”, Proceedings of IEEE East-West Design & Test Symposium (EWDTS’2015), 2015, pp. 264-267.

[6] A. Ling, D.P. Singh, S.D. Brown, “FPGA technology mapping: a study of optimality”, DAC '05 Proceedings of the 42nd annual Design Automation Conference, 2005, pp. 427-432. DOI: 10.1145/1065579.1065693.

[7] Y. Li, E. Tsai, M. Perkowski, X. Song, “Grover-based ashenhurst-curtis decomposition using quantum language quipper”, Quantum Information & Computation archive Volume 19 Issue 1-2, February 2019, pp. 35-66.

[8] M. A. Perkowski and S. Grygiel, “A survey of literature on function decomposition,” Portland State University, Tech. Rep. Ver. IV, 1995, p. 188.

[9] H.A. Curtis, “Generalized Tree Circuit - The Basic Building Block of an Extended Decomposition Theory", J. Assn. Comput. Mach, 1963, pp. 562-581.

[10] S.V. Gavrilov, A.L. Glebov, “BDD-based circuit level structural optimization for digital CMOS”, 1-st Intern. Workshop «Multi-Architecture Low Power Design», 1999, P. 45–49.

[11] S.V. Gavrilov, D.I. Ryzhova, N.O. Vasilyev, “Models and methods of inter-gate resynthesis at the transistor level for nanoelectronic circuits based on FinFETs”, 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, 2018, pp. 1364-1367. DOI: 10.1109/EIConRus.2018.8317350.

[12] S. Gavrilov, D. Zheleznikov, V. Khvatov, R. Chochaev, “Clustering Optimization Based on Simulated Annealing Algorithm for Reconfigurable SystemsOn-Chip”, 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2018, pp. 1492-1495. DOI: 10.1109/EIConRus.2018.8317380.

[13] M. Nachtigall, P. Ferreira, F. Marques, “Simulated Annealing applied to LUT-based FPGA Technology Mapping”, 2017 Sixteenth Mexican International Conference on Artificial Intelligence (MICAI), 2017, pp. 23-29. DOI: 10.1109/MICAI-2017.2017.00012/

[14] J. Cong and Yuzheng Ding, "FlowMap: an optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 13, no. 1, 1994, pp. 1-12,. DOI: 10.1109/43.273754. J. Cong and Yuzheng Ding, "FlowMap: an optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 13, no. 1, 1994, pp. 1-12,. DOI: 10.1109/43.273754