УДК 004.023

Алгоритм глобального размещения структурированных заказных схем

А.Е. Андреев[1], И. Пависич[1], А.С. Русаков[1,2]

[1]eASIC, [2]ИППМ РАН, rusakov@easic.com

***Аннотация* — Предложен алгоритм глобального размещения структурированных заказных схем. Алгоритм состоит из двух этапов. На первом этапе используется метод последовательного размещения, на втором метод отжига. Разработанные алгоритмы последовательного размещения и сбалансированного разбиения графа схемы позволяют эффективно расположить ячейки схемы с учетом пользовательских ограничений на позиции ячеек, временные задержки, и учитывают множественные ресурсы структурированной схемы.**

***Ключевые слова* — размещение СБИС, минимальное сечение.**

I. Введение

В последнее время структурированные матричные кристаллы, программируемые производителем (структурированные заказные микросхемы), снова заняли определенную нишу на рынке. Примером структурированного матричного кристалла могут служить микросхемы семейства Nextreme-2 [1]. В структурированных заказных схемах (structured ASIC) обычно все маски, кроме одной, заранее изготовлены. Программирование схемы осуществляется один раз на фабрике с помощью одной маски. Время проектирования, количество и стоимость масок на порядок ниже, чем для заказных схем (ASIC). По сравнению с ПЛИС, где для программирования кристалла используются дополнительные транзисторы, показатели скорости, энергопотребления и цены каждого кристалла для структурированных заказных схем оказываются лучше.

Одним из ключевых этапов маршрута проектирования структурированных схем является размещение элементов схемы на кристалле. Обычно задачу размещения решают в несколько этапов, одним из которых является глобальное размещение. На этом этапе определяются приближенные координаты ячеек, следующими этапами являются физический ресинтез, детальное размещение, упаковка, локальная оптимизация. Неудачное глобальное размещение не позволяет достичь оптимальных параметров схемы, таких, как быстродействие и энергопотребление.

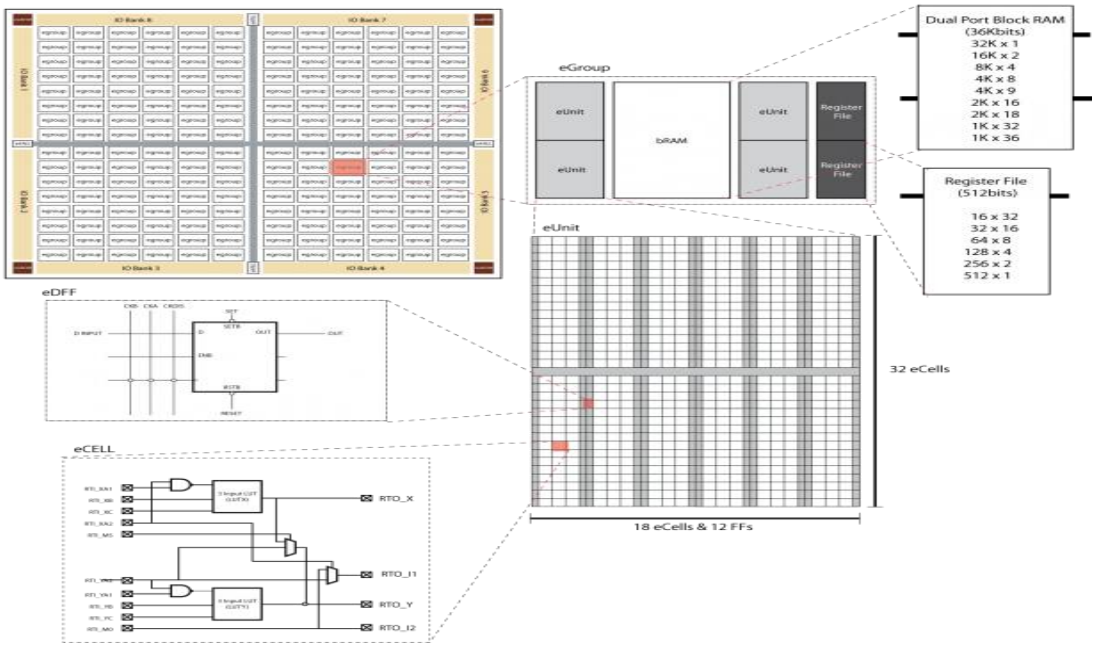

Рис. 1. Схема прибора Nextreme-2 [1]. Прибор состоит из матрицы групп элементов (eGroups), каждая группа состоит из 4 eUnits, одного элемента памяти Bram и двух регистровых файлов. Каждый eUnit содержит 32*18 ячеек логики eCells и 32*12 регистров

Размещению структурированных заказных схем 90нм технологии Nextreme посвящена работа [5]. Однако в ней не учитываются ограничения на временные задержки цепей, используется так называемая “виртуальная платформа”, т.е. из области кристалла вычитаются подобласти, где может быть размещена память, и размещение логики происходит на меньшей области без учета запрещенных позиций. Такая “виртуальная платформа” существенно влияет на результат глобального размещения и такое упрощение желательно исключить.

Для размещения элементов структурированной заказной схемы необходимо учитывать матричную структуру кристалла, похожую на структуру ПЛИС. Например, в 45нм технологии Nextreme-2 [1], кристалл разбит на группы элементов, каждая из которых состоит из логических элементов - eCell, регистров и блоков памяти (см. рис. 1). Электрическая схема для Nextreme-2 может содержать миллионы элементов, и использование известных программ размещения ПЛИС, например, [10], требует больших вычислительных ресурсов [2].

Аналитические методы оптимизации размещения, популярные при размещении заказных схем, для нашей задачи оказываются менее эффективными, так как области размещения памяти являются недоступными для других ячеек и задача становится существенно не выпуклой. Использование метода имитации отжига [9, 10] является стандартным подходом для размещения элементов на ПЛИС. В методе отжига можно учесть дискретную матричную структуру прибора, совместно оптимизировать длину проводов и задержки схем. Изменения в размещении элементов всегда инкрементальны, поэтому можно достаточно точно оценивать критичность временных путей, что гораздо сложнее сделать в алгоритме последовательного размещения. Очевидным недостатком метода отжига является его длительное время работы. Для схем с большим количеством элементов метод отжига не успевает за приемлемое время найти оптимальное размещение, хотя локально размещение близко к оптимальному.

Чтобы улучшить результаты глобального размещения мы разработали вариант метода последовательного размещения (mincut placement). Этот метод широко применяется для оптимизации размещения заказных схем (см., например, [3, 6]). Предложенная процедура глобального размещения состоит из двух этапов. На первом этапе мы запускаем алгоритм последовательного размещения. На последнем этапе используется метод отжига, который фактически решает задачу оптимизации размещения ячеек в локальных подобластях кристалла и минимизации сравнительно небольшого числа критических путей, что может быть сделано с гораздо меньшими затратами.

На базе подходов многоуровнего разбиения графа, примененных в [4, 8], нами разработан новый алгоритм сбалансированного разбиения графа схемы, который позволяет, учитывая матричную структуру кристалла, эффективно находить сильно связанные компоненты в схеме и располагать их на кристалле, минимизируя длину проводов.

II. Алгоритм последовательного размещения СБИС

Схема представляется в виде гиперграфа *H(V, E),* где *V*—множество ячеек схемы, *E* - множество ребер, описывающих цепи в схеме. Для каждого ребра и каждой вершины определен вес $w(e_i)$, $w(v_i)$. Весом вершины мы называем вектор $w(v_i) = \left(w_i^{\,1},...,w_i^{\,N}\right)$, где N – число ресурсов кристалла, а $w_i^{\,k}$ - площадь k-го ресурса, занимаемая иерархической ячейкой. В случае плоского описания схемы все веса, кроме одного, равны 0.

Длина цепи $P(e_i)$ оценивается как полупериметр прямоугольника, ограничивающего все контакты цепи. Классическая задача размещения элементов схемы состоит в том, чтобы минимизировать взвешенную сумму длин с весами ребер $w(e_i)$:

$$HPLW = \sum_i w(e_i) * P(e_i),\ i = 1,...,|E|, \quad (1)$$

расставив ячейки по подобластям так, чтобы в каждой подобласти сумма площадей ячеек не превышала допустимую площадь для данного типа ресурса. Одним из подходов к решению этой задачи является алгоритм последовательного размещения схемы или размещения “сверху-вниз”. В этом алгоритме область кристалла делится на две подобласти и с помощью алгоритмов поиска минимального сечения в гиперграфе граф схемы разбивается на подграфы, ассоциированные с центром каждой подобласти так, чтобы выполнялось ограничение на плотность элементов. Так повторяется рекурсивно для каждой области и для каждого подграфа, пока размер областей достаточно большой или число ячеек в подграфе меньше некоторого числа. Важно отметить, что в отсутствие внешних терминалов оптимизация сечения графа ведет непосредственно к оптимизации длины проводников в двух подобластях.

Как и в классическом алгоритме мы чередуем вертикальное и горизонтальное разбиения разбиваемых областей. Так как исходная область размещения на кристалле близка к квадрату, такой порядок разбиения является оптимальным. Для поиска оптимального сечения мы используем алгоритм сбалансированной бисекции графа, описываемый в следующей секции. При этом учитываются ограничения на площадь для каждого из 4 типов ячеек (eCells, eDFF, Bram, Regfiles). При

разбиении подобласти на две области *A* и *B* допускается 5% превышение площади ячеек над доступной площадью. После того, как разбиение графа найдено, мы сдвигаем границу *A* и *B* так, чтобы ограничение на площадь выполнялось.

Для оптимизации длины проводников при разделении текущей области кристалла необходимо предсказывать позиции ячеек, находящихся вне данной области. В англоязычной литературе эта процедура называется “terminal propagation”. В нашей работе для определения позиций внешних ячеек и их контактов (терминалов) на каждом уровне иерархии разбиения кристалла используется две итерации. На первой итерации мы считаем, что ячейки расположены в центрах, ассоциированных с этими ячейками подобластей кристалла. На второй итерации предполагается, что терминалы расположены в центрах разбитых на первой итерации подобластей. Очевидно, что большее количество таких итераций позволяет улучшить конечную длину проводов, но в нашем случае численные эксперименты показали, что 2 итерации являются хорошим компромиссом между временем работы и качеством размещения.

Для того чтобы приблизить задачу минимизации сечения графа к задаче минимизации длины проводников мы используем предложенный в [3] подход модификации весов ребер с терминалами. В этом подходе веса умножаются на число, изменяющееся от 0.5 до 1 так, чтобы изменение взвешенного сечения графа в точности соответствовало изменению функционала взвешенной длины цепей.

Алгоритм последовательного размещения усложняется, если заданы пользовательские ограничения на возможные позиции некоторых ячеек. Т.е. пользователь задает допустимый регион R_j на кристалле и множество ячеек $I_j(i_{1j},...,i_{Nj})$. Требуется, чтобы в результате ячейки оказались внутри своих регионов. Для этого при разбиении области на две необходимо учитывать следующие ситуации:

- регион пересекает обе подобласти. В таком случае при бисекции графа необходимо добавить условие, что плотность ячеек, принадлежащих региону, не нарушена в обоих подобластях.
- Регион пересекает одну подобласть. В этом случае мы можем исключить ячейки региона из разбиваемого графа и расположить их в центр пересечения области с регионом.
- Регион не пересекается с нашей областью, или ячейка региона не принадлежит разбиваемой подобласти, но одна из цепей связана с этой ячейкой. Тогда, если размер региона меньше, чем текущий размер разбиваемых подобластей, мы можем предположить, что ячейки из региона и внешние терминалы цепей находятся в центре региона.

A. Размещение элементов памяти схемы

В технологии Nexteme-2 поддерживаются два типа элементов памяти, Blocked RAM и регистровые файлы. Размер элементов памяти больше, чем размер подобластей на последних уровнях иерархии алгоритма размещения. Количество самих элементов памяти и их допустимых позиций сравнительно невелико, порядка 1000. На первых уровнях алгоритма мы держим их в графе схемы, но считаем отдельным ресурсом со своим допустимым значением в каждой подобласти. Начиная с выбранного уровня иерархии метода последовательного разбиения, мы используем специальную процедуру размещения памяти, которая считает остальные ячейки зафиксированными в их текущих позициях в центрах подобластей. Эта процедура оптимизирует размещение ячеек Bram и RegFiles, фиксирует их, после чего мы можем считать их контакты внешними терминалами с известными координатами. Процедура запускается после каждой итерации алгоритма последовательного разбиения.

III. Алгоритм сбалансированного разбиения графа

Задача нахождения сбалансированного минимального сечения гиперграфа электрической схемы в нашем случае формулируется следующим образом: необходимо найти такое разбиение вершин на два подмножества *A*, *B*, чтобы минимизировать сумму весов ребер, принадлежащих сечению графа Ψ:

$$Cut = \sum_{e_i \in \Psi} w(e_i), \tag{2}$$

при условии, что не нарушено ограничение на доступные ресурсы *W(A)* и *W(B)* в областях *A* и *B*:

$$\begin{aligned} &\sum_{v_i \in A} w^k(v_i) \le W^k(A), \\ &\sum_{v_i \in B} w^k(v_i) \le W^k(B), k = 1,..., N. \end{aligned} \tag{3}$$

Если пользователь задал дополнительные ограничения на размещение ячеек, и области *A* и *B* имеют не пустое пересечение с пользовательским регионом R_j, то появляется дополнительное условие на плотность ячеек принадлежащих региону:

$$\begin{aligned} &\sum_{v_i \in A, v_i \in I_j} w^k(v_i) \le W^k(R_j \times A), \\ &\sum_{v_i \in B, v_i \in I_j} w^k(v_i) \le W^k(R_j \times B), k = 1,..., N. \end{aligned} \tag{4}$$

Задача сбалансированного разбиения графа, как известно, NP-полная. Для этой задачи мы разработали алгоритм бисекции графа на базе подхода многоуровнего разбиения, разработанного в [4]. В работе [8] проведен детальный анализ применения этого подхода для задач, возникающих при размещении СБИС.

Наш алгоритм состоит из нескольких этапов:

- кластеризация гиперграфа (coarsening);
- нахождение нескольких лучших первоначальных разбиений кластеризованного графа (initial partition);
- развертывание кластеризованного графа и оптимизация разбиения на каждом уровне для выбранных решений (refinement), выбор лучшего решения.

A. Кластеризация графа

Как показано в [4], кластеризация необходима для качественного решения задачи поиска минимального сечения. Кластеризацией гиперграфа называется процесс построения гиперграфа, содержащего меньшее число узлов, чем исходный. К полученному графу опять применяется процедура кластеризации, и в итоге мы получаем иерархию гиперграфов, состоящую из нескольких уровней.

В нашей работе мы реализовали модификации известных алгоритмов HEC (heavy edge coarsening), MHEC (modified heavy edge coarsening) [4], HEM (heavy edge matching) [8], FC (first choice).

При сравнении алгоритмов кластеризации мы обнаружили, что алгоритмы HEC и MHEC достаточно быстро достигают насыщения. Размерность кластеризованного гиперграфа оказывается слишком большой, и поиск минимального сечения такого графа оказывается неэффективным. Чтобы улучшить вычислительную эффективность алгоритма, к гиперграфу, полученному из HEC или MHEC, мы применяем метод кластеризации FC, который успешно сжимает гиперграф еще в несколько раз.

В отличие от работы [8] мы не используем случайный обход ячеек гиперграфа. Эксперименты показывают, что упорядочивание ячеек в алгоритмах HEM, FC в среднем существенно улучшает качество разбиения. Первыми мы выбираем узлы с наименьшим количеством связей. Если число связей одинаковое, то приоритет имеют узлы с меньшим весом. Классические алгоритмы сортировки требуют *O(Nlog(N))* операций. Чтобы не ухудшить сложность кластеризации мы используем приближенную сортировку с помощью многоуровнего алгоритма “сортировки вычерпыванием” (bucket sort). Недостатком упорядочивания является то, что при детерминированном порядке обхода мы теряем возможность запускать множество раз один и тот же алгоритм и выбирать лучшее решение, как реализовано в [4,8]. Вместо этого мы используем в нескольких запусках разные алгоритмы кластеризации. Все этапы повторяются для каждого из выбранных алгоритмов кластеризации и выбирается решение с наименьшим сечением.

При кластеризации мы запрещаем объединяться узлам графа, принадлежащим разным регионам. Также за исключением регистров и логических ячеек не допускается кластеризация узлов, соответствующих разным типам ресурсов.

B. Первоначальное разбиение гиперграфа

Применяемый алгоритм нахождения минимального разреза плоского гиперграфа аналогичен разработанной в [8] модификации метода Fiduccia-Mattheyeses (FM). Этот метод на входе имеет некоторое первоначальное разбиение (например, случайное) и итеративно улучшает его. Сложность алгоритма - $O(|V|)$. Результат существенно зависит от входного разбиения, поэтому из множества решений, полученных для некоторого набора первоначальных разбиений, в результате выбирается лучшее. В наших экспериментах такой набор состоял из 35 вариантов для каждого из используемых методов кластеризации. Такое большое число вариантов выбрано на основе анализа поведения алгоритма на наиболее важных для нас дизайнах. Основным отличием от [8] является то, что мы допускаем только те движения ячеек, которые не нарушают ни одно из условий баланса (3)-(4). Кроме того, мы не удаляем из “корзины движений” (gain bucket) ячейки, которые на данном шаге не сбалансированы, и пытаемся их передвигать в другую подобласть как только условие баланса будет достигнуто. Эта модификация ухудшает быстродействие алгоритма, но в наших экспериментах мы обнаружили, что игнорирование временно несбалансированных движений ячеек может очень сильно увеличить размер сечения. Число итераций FM при первоначальном разбиении кластеризованного гиперграфа - не более 80. В действительности, обычно достаточно 5-6 итераций. Мы также чередуем итерации FM и CLIP модификацию FM алгоритма. При декластеризации графа мы используем не более 5 итераций.

C. Развертывание графа

Так как метод FM, используется в рамках многоуровнего алгоритма поиска минимального сечения, при оптимизации сечения кластеризованного графа обычно выбирается не одно решение, а несколько лучших (в наших расчетах 10). Эти несколько решений оптимизируются при развертывании графа также с помощью метода FM, но с начальным приближением, определенным решением на более высоком уровне иерархии кластеризованных графов.

D. Сравнение на тестах ISPD 98

В таблице 1 мы приводим сравнение результатов нашего алгоритма поиска минимального сечения с алгоритмами [4, 8]. Сравнение проведено на общедоступных тестах ISPD98. Допускаемое превышение над суммой площадей – 2%. Также мы приводим величины сечения при использовании разных методов кластеризации по сравнению с выбором лучшего решения из 4 методов кластеризации. При этом суммарное число начальных приближений и решений, улучшаемых при развертывании графа, одинаковое во всех пяти случаях.

Таблица 1

Величина сечения графа для нескольких алгоритмов кластеризации и при выборе лучшего решения. В двух последних столбцах приведены результаты [4, 8], взятые с http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/Partitioning/HMetisML02Tab.html

	Выбор лучшего	HEC	MHEC	HEM	FC	MLPART	HMETIS
ibm01	233	289	289	345	248	227	245
ibm02	269	357	356	295	351	294	299
ibm03	763	799	763	784	738	818	855
ibm04	502	664	607	507	554	531	534
ibm05	1739	1745	1789	1748	1753	1750	1741
ibm06	525	585	508	612	538	564	605
ibm07	839	856	860	839	857	793	794
ibm08	1168	1189	1194	1180	1168	1206	1208
ibm09	546	531	548	548	538	527	524
ibm10	1119	1119	1161	1119	1124	1211	1193
ibm11	821	833	825	933	821	842	813
ibm12	2089	2186	2188	2109	2121	2353	2131
ibm13	1050	1063	931	1223	1202	1036	931
ibm14	1792	1840	1790	1903	2285	1860	1865
ibm15	2494	2237	2285	2650	2608	2243	2221
ibm16	1739	2525	2899	1834	2221	1853	1713
ibm17	2376	2352	2337	2468	2463	2353	2460
ibm18	1776	1959	1895	1860	1776	1737	1706
TOTAL	**21840**	**23129**	**23225**	**22957**	**23366**	**22198**	**21838**

IV. Оптимизация задержек

Оптимизация задержек в алгоритмах последовательного размещения на базе бисекции графа достаточно трудная задача. В этих алгоритмах при переходе с одного уровня на другой могут измениться почти все пути, в отличие от алгоритмов, в которых размещение улучшается инкрементально, поэтому предсказать критичность пути до разбиения графа оказывается практически невозможно. Именно необходимость оптимизации задержек в схеме не позволяет нам отказаться от использования метода отжига на последнем этапе размещения. Метод отжига достаточно хорошо минимизирует задержки на небольшом количестве временных путей, поэтому мы формулируем наш алгоритм последовательного размещения так, чтобы уменьшить сумму задержек, оставляя минимизацию критических путей на последний этап. Для решения этой задачи мы предварительно для каждого ребра гиперграфа e_i рассчитываем "допустимую длину" $Ladm_i$ такую, что все временные пути, проходящие только через ребра, длина которых меньше их допустимой длины, удовлетворяют временным ограничениям. Далее для каждой цепи на выбранном уровне иерархии размещения мы можем оценить ее возможную максимальную $L\max_i$ и минимальную $L\min_i$ длину (см. [7]). Тогда мы можем изменить вес каждого ребра графа в соответствии со следующей формулой:

$$w = \begin{cases} 1, \ \textit{если} \ \ L\max_i < Ladm_i \\ w_{critical}, \ \textit{если} \ \ \ L\min_i > Ladm_i \\ w_{critical} * \dfrac{(L\max_i - Ladm_i)}{L\max_i - L\min_i} \end{cases},$$

где $w_{critical}$ - константа. В наших экспериментах мы выбрали $w_{critical} = 1.5$.

V. Численные эксперименты

Эффективность разработанных алгоритмов оценивалась на задаче размещения электронных схем на кристаллах Nextreme-2 eASIC. В таблице 2 приведено значение средней длины проводников после глобального размещения. В одном случае мы в качестве начального для метода отжига использовали случайное размещение и потом улучшали его несколькими итерациями алгоритма аналитического размещения. Во втором случае мы использовали на первом этапе разработанный алгоритм последовательного размещения. Для всех тестов мы получили улучшение качества глобального размещения, которое также влияет на итоговые характеристики схемы.

Таблица 2

Длина проводников после глобального размещения без использования метода последовательного размещения и с ним

Средняя длина в микронах после глобального размещения			
	Initial Placement and SA	MinCut and SA	Число ячеек
test1	159	138	520000
test2	164	112	950000
test3	124	98	480000
test4	100	75	412000
test5	147	114	830000

VI. Заключение

Разработан новый алгоритм глобального размещения структурированных заказных схем. Совместное использование метода последовательного разбиения и метода отжига позволяет улучшить как длину межсоединений, так и временные характеристики размещенной схемы. Для эффективного решения задачи сбалансированного разбиения графа схемы разработан новый иерархический алгоритм разбиения гиперграфа, который позволяет учесть как ограничения на количество свободных позиций определенного типа в подобластях кристалла, так и пользовательские ограничения на позиции ячеек. Разработанные алгоритмы показали свою высокую эффективность для 45нм технологии Nextreme-2 фирмы eASIC.

Литература

[1] “eASIC Corporate Website”, http://www.easic.com, 2012.

[2] Schmit H., Gupta A., Ciobanu R. Placement Challenges for Structured ASICs // in ISPD ’08: Proceedings of the 2008 International Symposium on Physical design. 2008. P. 84–86.

[3] Selvakkumaran N. and Karypis G. Theto—A fast and high-quality partitioning driven global placer, Dept. Comput. Sci. Eng., Univ. Minnesota, Minneapolis, MN, Tech. Rep. 03-46, Nov. 2003.

[4] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. 1997. Multilevel hypergraph partitioning: application in VLSI domain // In Proceedings of the 34th annual Design Automation Conference (DAC '97). ACM, New York, NY, USA. 1997. P. 526-529.

[5] Ahutosh Chakraborty, Anurag Kumar, and David Z. Pan. RegPlace: a high quality open-source placement framework for structured ASICs // In Proceedings of the 46th Annual Design Automation Conference (DAC '09). ACM, New York, NY, USA. 2009. P. 442-447.

[6] J. A. Roy, D. A. Papa, S. N. Adya, H. H. Chan, A. N. Ng, J. F.Lu, and I. L. Markov, “Capo: Robust and Scalable Open-source min-cut Floorplacer,” in ISPD ’05: pp. 224–226, ACM,2005

[7] T. Gao, P. M. Vaidya, and C. L. Liu. 1992. A performance driven macro-cell placement algorithm. In Proceedings of the 29th ACM/IEEE Design Automation Conference (DAC '92). IEEE Computer Society Press, Los Alamitos, CA, USA, 147-152.

[8] Papa D.A. and Markov I.L. Hypergraph Partitioning and Clustering // in Approximation Algorithms and Metaheuristics, T. Gonzalez, ed.; pages 61-1- 61-19, CRC Press, CRC Press, 2007.

[9] Sechen, C. VLSI Placement and Global Routing Using Simulated Annealing. KluwerAcademic Publishers, Boston, Ma., 1988.

[10] Vaughn Betz and Jonathan Rose, VPR: A New Packing, Placement and Routing Tool for FPGA Research // International Workshop on Field Programmable Logic and Applications. 1997. P. 213-222.