19. *Kostyukov V.A., Medvedev M.Yu., Pshikhopov V.Kh.* Algoritmy planirovaniya sglazhennykh individual'nykh traektoriy dvizheniya nazemnykh robotov [Algorithms for planning smoothed individual trajectories of ground robots], *Mekhatronika, avtomatizatsiya, upravlenie* [Mechatronics, Automation, Control], 2022, Vol. 23 (11), pp. 585-595.
20. *Kostjukov V. Pshikhopov V., Medvedev M.* Optimization of mobile robot movement on a plane with finite number of repeller sources, *SPIIRAS Proceedings,* 2020, Vol. 19 (1), pp. 43-78.
21. *Zabarankin M., Uryasev S., Pardalos P.* Optimal Risk Path Algorithms, *Cooperative Control and Optimizaton. Dordrecht: Kluwer Acad., 2002,* pp. 271-303.

Статью рекомендовал к опубликованию д.т.н., профессор В.В. Курейчик.

Костюков Владимир Александрович – НКБ «РиСУ»; e-mail: wkost-einheit@yandex.ru; г. Таганрог, Россия; тел.: 88634371694; к.т.н.; с.н.с.

Медведев Михаил Юрьевич – НИИ робототехники и процессов управления Южного федерального университета; e-mail: medvmihal@sfedu.ru; г. Таганрог, Россия; тел.: 88634371694; д.т.н.; в.н.с.

Пшихопов Вячеслав Хасанович – e-mail: pshichop@rambler.ru; тел.: 88634371694; д.т.н.; профессор; директор.

Kostyukov Vladimir Aleksandrovich – Joint-Stock Company "Robotics and Control Systems"; e-mail: wkost-einheit@yandex.ru; Taganrog, Russia; phone: 88634371694; can. of eng. sc.; senior researcher.

Medvedev Mikhail Yurjevich – R&D Institute of Robotics and Control Systems; e-mail: medvmihal@sfedu.ru; Taganrog, Russia; phone: 88634371694; dr. of eng. sc.; leading researcher.

Pshikhopov Viacheslav Khasanovich – e-mail: pshichop@rambler.ru; phone: 88634371694; dr. of eng. sc.; professor; director.

УДК 621.3.049.771.14 DOI 10.18522/2311-3103-2023-5-48-57

Д.Б. Шокарев, Р.Ж. Чочаев, А.Н. Щелоков, С.В. Гаврилов

РАЗРАБОТКА АЛГОРИТМА ДЕТАЛЬНОГО РАЗМЕЩЕНИЯ НА ПЛИС

Иерархические программируемые логические интегральные схемы (ПЛИС) состоят из множества логических блоков, объединенных в группы. Для успешной трассировки необходимо оптимальное размещение элементов в пределах групп с учётом особенностей архитектуры локальных связей. Классические алгоритмы не способны обеспечить учёт различных особенностей архитектуры. Решение данной проблемы возможно только путем разработки специализированных алгоритмов. В данной работе представлен алгоритм детального размещения, в котором для вычисления оптимальных позиций элементов в группе была разработана новая метрика, позволяющая оценить количество доступных локальных связей между элементами в группах логических блоков с учётом особенностей архитектуры связей между ними. Алгоритм детального размещения состоит из нескольких этапов. На первом этапе группа логических элементов представляется в виде ориентированного графа. На втором этапе определяется порядок размещения логических элементов в группе с помощью алгоритма поиска в ширину. На финальном этапе для каждого элемента, согласно полученному порядку, определяется оптимальное размещение в группе с учётом разработанной метрики. Если среди свободных позиций для размещения в группе нет оптимальной, то проверяются занятые позиции. Текущий элемент назначается на занятую позицию, а для замененного элемента выполняется поиск новой. Такая замена может проводиться многократно, увеличивая вероятность нахождения оптимальной конфигурации. Предложенный алгоритм был реализован и протестирован на наборах тестовых схем. На основе результатов тестирования выполнено сравнение представленного алгоритма с алгоритмом последовательного размещения. Сравнение алгоритмов показало, что приме-

нение разработанного алгоритма в маршруте проектирования в базисе специализированной ПЛИС позволяет сократить в среднем на 10% количество задействованных в трассировке глобальных коммутационных шин и увеличить количество используемых локальных трассировочных ресурсов в среднем на 30%. Полученные результаты подтверждают работоспособность алгоритма и доказывают, что внедрение учета архитектуры внутренних связей ПЛИС повышает эффективность использования доступных трассировочных ресурсов.

Размещение; автоматизация проектирования; ПЛИС.

D.B. Shokarev, R.Zh. Chochaev, A.N. Schelokov, S.V. Gavrilov

DEVELOPMENT OF THE DETAILED PLACEMENT ALGORITHM FOR FPGAS

Hierarchical field-programmable gate arrays (FPGAs) consist of an array of programmable logic blocks arranged into groups. Successful routing requires optimal placement of logic elements within the groups, considering the architectural features of the local interconnections. Classical algorithms are not able to consider these features. That’s why, the development of new algorithms is required. In this paper, we present a detailed placement algorithm with a new metric that allows us to estimate the number of available local interconnections inside the groups of logic blocks, considering the architectural features of the local interconnections. The detailed placement algorithm consists of several stages. At the first stage, the group of logic elements is transformed into a directed graph. Then, the placement order of logic elements in the group is determined using the breadth-first search algorithm. At the final stage, for each element, according to the obtained order, the optimal placement in the group is determined, considering the new metric. If there is no optimal position in the group among the free ones, the occupied positions are checked. The current element is placed in the occupied position, and a new position is searched for the replaced element. Such replacements can be performed repeatedly, increasing the probability of finding the optimal placement configuration. The proposed algorithm was verified on a set of benchmark circuits ISCAS’85, ISCAS’89, Cpu8080 and VGA. Experimental results show that the developed algorithm reduces the number of global interconnections used for global routing by 10% on average and increases the number of local interconnections used for detailed routing by 30% on average compared to the sequential placement algorithm. The average routing time remained unchanged.

Placement; electronic design automation (EDA); FPGA.

Введение. Программируемые логические интегральные схемы (ПЛИС) – это цифровые микросхемы, логика которых задаётся посредством программирования (проектирования) [1]. В данной работе рассматривается специализированная ПЛИС, обладающая классической иерархической архитектурой, описанной в работах [2–5]. Она состоит из множества логических блоков (ЛБ), объединенных в группы логических блоков (ГЛБ) по 16 ЛБ. Каждый ЛБ состоит из двух основных частей: таблицы преобразования (англ. *look-up table,* LUT), реализующей таблицу истинности логической функции, и триггера.

В иерархической архитектуре рассматриваемой ПЛИС существует разделение трассировочных ресурсов на локальные (в пределах ГЛБ) и глобальные (между ГЛБ). Из-за этого соединение всех ЛБ в пределах ГЛБ осуществляется через коммутатор, описание которого приведено в статье [4]. Коммутатор связывает между собой три типа шин: глобальные шины (ГШ), соединяющие различные ГЛБ, шины локальной обратной связи (ШЛОС), соединяющие ЛБ одной группы, и цепи данных, соединенные непосредственно с входными терминалами ЛБ.

Одной из особенностей архитектуры рассматриваемой ПЛИС является нестандартное устройство коммутаторов. В классических ПЛИС в коммутаторы закладывается возможность создания связи между любыми выходами и любыми входами ЛБ одной группы [6]. Такой подход дает максимальную гибкость, но имеет серьезный недостаток – неэффективное использование площади кристалла, так как одновременно может использоваться лишь часть заложенных трассировочных ресурсов. В рассматриваемой ПЛИС количество связей между ЛБ одной группы было значительно снижено путем редукции связей между парами ЛБ таким образом, что выход каждого ЛБ может быть соединен либо с первым или третьим, либо со вторым или четвертым входами других ЛБ той же группы.

Данное решение позволяет сократить площадь коммутаторов, но при отсутствии учета архитектуры связей лишает возможности реализовать часть локальных связей в пределах ГЛБ и приводит к неоптимальному использованию глобальных коммутационных шин. В результате наблюдается рост задержки между соседними ЛБ и перегрузка соединений между ГЛБ.

В маршруте топологического проектирования в базисе иерархических ПЛИС этап размещения состоит из трех шагов [7]. На первом шаге выполняется кластеризация логических элементов (ЛЭ) проектируемой схемы в группы [8–9]. На втором шаге вычисляется глобальное размещение полученных групп логических элементов [10-12], т.е. каждой группе назначается ГЛБ в ПЛИС. На финальном шаге выполняется детальное размещение [13–14], т.е. логические элементы в группах назначаются на конкретные ЛБ в пределах выбранной ГЛБ. В данной работе представлен алгоритм детального размещения, в основе которого лежит метрика качества размещения, учитывающая архитектуру связей внутри ГЛБ.

Последовательный алгоритм детального размещения. В данном разделе описывается последовательный алгоритм, с которым выполнено сравнение разработанного алгоритма. Последовательный алгоритм не имеет каких-либо принципиальных особенностей, но полезен в случаях, когда архитектура соединений ПЛИС предусматривает наличие прямых последовательных связей между соседними ЛБ в обход коммутатора. Его ключевая идея, изложенная в работе [15], заключается в размещении наиболее длинных цепочек элементов, чтобы максимально задействовать прямые последовательные связи между соседними ЛБ. Однако, в рассматриваемой ПЛИС такие связи отсутствуют, поэтому его можно рассматривать как базовый алгоритм без учета структуры межсоединений.

Последовательный алгоритм детального размещения состоит из трех этапов: инициализации, поиска наибольшей последовательности связанных элементов и размещения найденной последовательности. При этом последние два этапа повторяются до тех пор, пока не будут размещены все ЛЭ.

На первом этапе работы алгоритма группа логических элементов, полученная после кластеризации схемы, представляется в виде ориентированного графа $G = (V, E)$, где V – множество вершин, представляющих ЛЭ, E – множество ребер, представляющих цепи между ЛЭ. Затем из полученного графа удаляются все ребра, инцидентные лишь одной вершине и представляющие цепи, проходящие через границы ГЛБ, и создаются списки *Path* и *MaxPath*, в которых будут храниться обнаруженные последовательности вершин. Для поиска наибольшей взаимосвязанной последовательности ЛЭ используется алгоритм Дейкстры, описанный в работе [16], который позволяет обнаружить наикратчайший путь от вершины-источника до каждой из вершин графа. Затем выполняется восстановление пути от самой дальней вершины до вершины-источника, после чего найденный путь сохраняется в списке *MaxPath*. Все вершины из списка *MaxPath* размещаются в ГЛБ и удаляются из графа. Поиск и размещение последовательностей ЛЭ повторяется до тех пор, пока $V \neq \emptyset$. Псевдокод данного алгоритма представлен на рис. 1.

Стоит отметить, что последовательный алгоритм, хоть и позволяет упростить будущую трассировку, но не лишен недостатков. Удаление из графа наибольшей последовательности вершин приводит к потенциальной блокировке нескольких меньших последовательностей, размещать которые было бы выгоднее, а также к сложности внедрения в него ограничений, которые должны учитываться в ходе работы.

```
Создать G, Path и MaxPath
Пока |V| > 0
    maxLen = -1
    Для ∀ v ∈ V
        length = Dijkstra (G, i, Path)
        Если length > maxLen
            maxLen = length
            MaxPath = Path
    Для ∀ v ∈ MaxPath
        Разместить v
        Удалить v из V
    Очистить Path и MaxPath
Удалить G, Path и MaxPath
```

Рис. 1. Псевдокод последовательного алгоритма детального размещения

Представление и учет архитектуры ПЛИС. Для того, чтобы представить архитектуру внутренних связей ГЛБ, удобно использовать матрицу смежности. Каждый элемент такой матрицы обозначает наличие (1) или отсутствие (0) связи между выходом i-го ЛБ и входом j-го ЛБ. В рассматриваемом случае у каждого ЛБ существует четыре входа, но доступны они попарно (первый и третий или второй и четвертый), поэтому для описания всех связей достаточно двух матриц: A – для второго и четвертого портов, и B – для первого и третьего.

Кроме того, каждая из матриц содержит дополнительный столбец, содержащий суммы элементов каждой строки S_i^a и S_i^b, представляющие собой потенциально доступное количество входных терминалов каждого типа для i-го ЛБ:

$$A = \left(\begin{array}{cccc|c} a_{0,0} & a_{0,1} & \cdots & a_{0,15} & S_0^a \\ a_{1,0} & a_{1,1} & \cdots & a_{1,15} & S_1^a \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{15,0} & a_{15,1} & \cdots & a_{15,15} & S_{15}^a \end{array}\right) \quad B = \left(\begin{array}{cccc|c} b_{0,0} & b_{0,1} & \cdots & b_{0,15} & S_0^b \\ b_{1,0} & b_{1,1} & \cdots & b_{1,15} & S_1^b \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b_{15,0} & b_{15,1} & \cdots & b_{15,15} & S_{15}^b \end{array}\right).$$

Метрика качества размещения. Для того, чтобы учесть архитектуру связей между ЛБ, необходимо оценить то, насколько будущая позиция каждого ЛЭ соответствует установленным ограничениям. В связи с этим вводится целевая функция *Cost*:

$$Cost^i = C_{place}^i + \alpha \cdot C_{out}^i,$$

где α – корректирующий коэффициент; C_{place}^i, C_{out}^i — составляющие целевой функции для i-го ЛЭ. C_{place}^i служит для учета будущего расположения элементов и вычисляется по следующей формуле:

$$C_{place}^i = C_{place}^a + C_{place}^b,$$

$$C_{place}^a = \begin{cases} {F_i}^a - S_i^a, {F_i}^a > S_i^a \\ 0, \end{cases}, \qquad C_{place}^b = \begin{cases} {F_i}^b - S_i^b, {F_i}^b > S_i^b \\ 0, \end{cases},$$

где ${F_i}^a$ – необходимое число вторых и четвёртых портов для подключения выхода i-го ЛЭ; ${F_i}^b$ – необходимое число первых и третьих портов для подключения выхода i-го ЛЭ.

Вторая составляющая C_{out}^i служит для учета расположения элементов в процессе работы алгоритма:

$$C_{out}^i = C_{out}^{\mathrm{a}} + C_{out}^{\mathrm{b}},$$

$$C_{out}^a = \sum_j \overline{a_{i,j}}, \quad C_{out}^b = \sum_j \overline{b_{i,j}},$$

где j – индексы уже размещенных ЛЭ, к которым подключены выходы i-го ЛЭ.

Алгоритм детального размещения с учетом связей между ЛБ. Суть разработанного алгоритма детального размещения заключается в предварительном определении наиболее оптимальных позиций всех ЛЭ с учетом связей между блоками с последующим их размещением на данные позиции. Определение оптимальных позиций происходит путем их перебора с пересчетом *Cost* для каждого ЛЭ. Если находится размещение с *Cost* = 0, то оно считается оптимальным и поиск заканчивается. Если же подобное размещение не удается найти, то выбирается такое размещение, значение *Cost* для которого минимально. Разработанный алгоритм состоит из четырех этапов: инициализации, определения порядка обхода элементов, определения оптимальных позиций для всех элементов и размещения элементов на оптимальные позиции.

Данный алгоритм основывается на том же графовом представлении схемы, что и последовательный. Однако в разработанном алгоритме не создаются списки для хранения цепочек элементов, а используются массивы *P* и *tempP*, выступающие в роли модели ГЛБ, где индекс элементов соответствует индексу будущей позиции.

После инициализации определяется порядок обхода вершин, для чего используется модифицированный алгоритм поиска в ширину. На первом шаге из числа ранее не посещенных вершин выбирается начальная вершина *s* с максимальным числом исходящих ребер. Затем с помощью алгоритма поиска в ширину вычисляется кратчайшее расстояние от начальной вершины до остальных вершин. Если после обхода остаются непосещенные вершины, то весь процесс повторяется, пока каждой вершине в графе не будет задан вес *w*, равный кратчайшему расстоянию от *s*.

При дальнейшей работе алгоритма вершины обходятся в порядке роста веса *w* и уменьшения количества исходящих ребер. В случае равного приоритета рассматривается два направления обхода: прямое и обратное.

Определение позиций происходит путем их перебора с пересчетом *Cost* для каждого взаиморасположения ЛЭ. Очередная вершина добавляется в массив, рассчитывается значение *Cost* для всех его членов и, если суммарное значение *Cost* = 0 или проверены все позиции, то лучшая конфигурация сохраняется, а в массив добавляется следующая вершина.

Сначала проверяются все свободные позиции, и если среди них нет оптимальной, то начинается проверка занятых. В таком случае новая вершина временно фиксируется на проверяемой позиции, и запускается поиск уже для старой вершины, начиная со свободных позиций. Такая операция может проводиться многократно, увеличивая вероятность нахождения оптимальной конфигурации. В данной работе использовалась глубина перебора равная трем.

После определения всех позиций, если *Cost* = 0, то выполняется размещение всех ЛЭ, в противном случае результирующая конфигурация сохраняется, а второй и третий этапы повторяются с обратным порядком обхода. Псевдокод алгоритма детального размещения с учетом связей между блоками представлен на рис. 2.

```
Создать G, P и tempP
flagINV = 0
globalMinCost = ∞
INV:
Пока ∃ непосещенные v ∈ V
        s = FindMaxVer (G)
        distanceCounter (G, s, flagINV)
        Пока ∃ v ∈ V \ tempP
                minCost = ∞
                iterVertex = findNextVertex (G, P, flagINV)
                Для ∀ i ∈ [0,15]
                        iterCost = checkPos (tempP, iterVertex, i)
                        Если iterCost = 0 и ∀ v ∈ tempP
                                Сохранить tempP
                                break
                        Если iterCost < minCost
                                Сохранить tempP
                                minCost = iterCost
minCost = CalculateCost(tempP)
Если minCost < globalMinCost
        P = tempP
        globalMinCost = minCost
Если globalMinCost ≠ 0 и flagINV = 0
        flagINV = 1
        goto INV
Для ∀ v ∈ P
        Разместить v
```

Рис. 2. Псевдокод алгоритма детального размещения с учетом связей между блоками.

Результаты тестирования. Представленные алгоритмы были протестированы на наборе тестовых схем ISCAS'85 [17], ISCAS'89[18], схемах VGA [19] и Cpu8080 [20]. Сравнение алгоритмов производилось по трем параметрам: количеству использованных шин локальной обратной связи, количеству использованных глобальных шин и времени, затраченному на трассировку.

Из результатов сравнения видно, что количество задействованных в трассировке шин локальной обратной связи возросло в среднем на 30% (табл. 1), использование же глобальных шин сократилось в среднем на 10% (табл. 2). Исходя из этого, можно сделать вывод, что использование алгоритма детального размещения с учетом связей между блоками позволяет значительно сократить используемое количество глобальных трассировочных ресурсов за счет использования шин локальной обратной связи.

Таблица 1

Результаты тестов. Количество использованных ШЛОС

Схема	Количество ЛЭ	Количество использованных шин локальной обратной связи		
		Последовательный	С учетом связей	Δ, %
c880	123	96	145	+51
c6288	919	1016	1232	+21
s15850	1363	1299	1591	+22
VGA	2177	1815	2318	+28
Cpu8080	2550	1881	2575	+37
s38417	4240	3839	4733	+23
Δavg, %				+30

Таблица 2

Результаты тестов. Количество использованных ГШ

Схема	Количество ЛЭ	Количество использованных глобальных шин		
		Последовательный	С учетом связей	Δ, %
c880	123	345	296	-14
c6288	919	1954	1738	-11
s15850	1363	3533	3241	-8
VGA	2177	6990	6487	-7
Cpu8080	2550	6980	6286	-10
s38417	4240	10401	9507	-9
Δavg, %				-10

Среднее время трассировки (таблица 3) не изменилось значительно, но имеет место существенный разброс значения параметра на различных схемах. Это стало результатом весомого усложнения трассировки в пределах ГЛБ для схем с высокой связанностью элементов, что, однако, приводит к большему росту эффективности использования трассировочных ресурсов, как видно на примере схемы Cpu 8080 [20].

Таблица 3

Результаты тестов. Время трассировки

Схема	Количество ЛЭ	Время последующей трассировки, сек		
		Последовательный	С учетом связей	Δ, %
c880	123	367	328	-11
c6288	919	787	829	+5
s15850	1363	4032	4380	+9
VGA	2177	9237	8133	-12
Cpu8080	2550	7625	9941	+30
s38417	4240	12975	12017	-7
Δavg, %				+2

Заключение. В данной работе представлен алгоритм детального размещения на основе метрики качества, учитывающей архитектуру связей между логическими блоками ПЛИС. Разработанный алгоритм состоит из четырех этапов: инициализации и определения порядка обхода группы размещаемых элементов с помощью алгоритма поиска в ширину, определения оптимальных позиций для всех элементов и их размещения. Для вычисления оптимальных позиций элементов внутри ГЛБ была разработана новая метрика, которая, в отличие от классических, таких как длина соединений и задержки, позволяет оценить количество доступных локальных связей между логическими элементами в ГЛБ.

Разработанный алгоритм и метрика оценки могут использоваться при проектировании в базисе отечественных ПЛИС. Экспериментальное сравнение разработанного алгоритма детального размещения с последовательным алгоритмом размещения показало, что применение разработанного алгоритма в маршруте проектирования в базисе рассматриваемой в работе ПЛИС позволяет сократить в среднем на 10% количество задействованных в трассировке глобальных коммутационных шин при росте количества используемых внутренних трассировочных ресурсов ГЛБ в среднем на 30%.

Финансирование. Работа выполнена при финансовой поддержке Минобрнауки в рамках государственного задания FSMR-2023-0002 (Соглашение №075-03-2023-024) и государственного задания FFNZ-2021-0001 (Соглашение №075-03-2023-247).

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Курский В.В., Сунка В.Я., Полынкова Е.В.* Программируемые ПЛИС // Машиностроение: республиканский межведомственный сборник научных трудов: по материалам Международной научно-технической конференции «Материалы, оборудование и ресурсосберегающие технологии в машиностроении», 06-09 апреля 2010 года: в 2 т. / под ред. Б.М. Хрусталева. – Минск: БНТУ, 2012. – Вып. 26, Т. 2. – С. 255-259.
2. *Farooq U., Marrakchi Z., Mehrez H.* Tree-based heterogeneous FPGA architectures: application specific exploration and optimization. – Springer Science & Business Media, 2012. – P. 186.
3. *Singh A., Parthasarathy G., Marek-Sadowska M.* Interconnect resource-aware placement for hierarchical FPGAs // IEEE/ACM International Conference on Computer Aided Design. ICCAD 2001. IEEE/ACM Digest of Technical Papers (Cat. No. 01CH37281). – IEEE, 2001. – P. 132-136.
4. *Boutros A., Betz V.* FPGA architecture: Principles and progression // IEEE Circuits and Systems Magazine. – 2021. – Vol. 21, No. 2. – P. 4-29.
5. ПЛИС емкостью 21 502 вентилей 5510ТС068. – АО «Микрон». – URL: https://mikron.ru/products/high-rel-ic/programmiruemaya-logika-fpga/fpga/product/5510ts068/ (дата обращения: 29.01.2022).
6. *Petelin O., Betz V.* Wotan: evaluating FPGA architecture routability without benchmarks // ACM Trans. Reconfigurable Technol. Syst. – 2018. – Vol. 11, No. 2. – P. 1-23.
7. *Фролова П.И., Чочаев Р.Ж., Иванова Г.А., Гаврилов С.В.* Алгоритм размещения с оптимизацией быстродействия на основе матриц задержек для реконфигурируемых систем на кристалле // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). – 2020. – Вып. 1. – С. 2-7.
8. *Фролова П.И., Хватов В.М., Чочаев Р.Ж.* Сравнительный анализ методов кластеризации и размещения схем для реконфигурируемых систем на кристалле // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). – 2022. – Вып. 4. – С. 63-70.
9. *Kureichik V.V., Zaporozhets D.Y., Zaruba D.V.* Partitioning of VLSI fragments based on the model of glowworm's behavior // Proceedings of the 19th International Conference on Soft Computing and Measurements, SCM 2016. – 2016. – P. 268-272.
10. *Li W. et al.* UTPlaceF 3.0: A parallelization framework for modern FPGA global placement // 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). – IEEE, 2017. – P. 922-928.

11. *Chen G., et al.* RippleFPGA: Routability-driven simultaneous packing and placement for modern FPGAs // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. – 2017. – Vol. 37, No. 10. – P. 2022-2035.
12. *Dhar S., et al.* Detailed placement for modern FPGAs using 2D dynamic programming // 2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). – IEEE, 2016. – P. 1-8.
13. *Nikolić S., Zgheib G., Ienne P.* Detailed Placement for Dedicated LUT-Level FPGA Interconnect // ACM Transactions on Reconfigurable Technology and Systems. – 2022. – Vol. 15, No. 4. – P. 1-33.
14. *Гаврилов С.В., Железников Д.А., Чочаев Р.Ж.* Разработка и сравнительный анализ методов решения задачи размещения для реконфигурируемых систем на кристалле // Известия высших учебных заведений. Электроника. – 2020. – Т. 25, № 1. – С. 48-57.
15. *Fan D.K., Shi P.* Improvement of Dijkstra's algorithm and its application in route planning // 2010 seventh international conference on fuzzy systems and knowledge discovery. – IEEE. 2010. – Vol. 4. – P. 1901-1904.
16. *Dijkstra E.W.* A note on two problems in connexion with graphs // Edsger Wybe Dijkstra: His Life, Work, and Legacy. – 2022. – P. 287-290.
17. *Bryan D.* The ISCAS'85 benchmark circuits and netlist format // North Carolina State University. – 1985. – Vol. 25. – P. 39.
18. *Brglez F., Bryan D., Kozminski K.* Combinational profiles of sequential benchmark circuits // 1989 IEEE International Symposium on Circuits and Systems (ISCAS). – IEEE, 1989. – P. 1929-1934.
19. 8080 Compatible CPU. – URL: https://opencores.org/projects/cpu8080
20. The OpenCores VGA/LCD Controller. – URL: https://opencores.org/projects/vga_lcd.

REFERENCES

1. *Kurskiy V.V., Sunka V.Ya., Polynkova E.V.* Programmiruemye PLIS [Programmable FPGAs], *Mashinostroenie: respublikanskiy mezhvedomstvennyy sbornik nauchnykh trudov: po materialam Mezhdunarodnoy nauchno-tekhnicheskoy konferentsii «Materialy, oborudovanie i resursosberegayushchie tekhnologii v mashinostroenii», 06-09 aprelya 2010 goda* [Mechanical engineering: republican interdepartmental collection of scientific papers: based on the materials of the International scientific and technical conference "Materials, equipment and resource-saving technologies in mechanical engineering", April 06-09, 2010]: In 2 vol., ed. by B.M. Khrustaleva. Minsk: BNTU, 2012, Issue 26, Vol. 2, pp. 255-259.
2. *Farooq U., Marrakchi Z., Mehrez H.* Tree-based heterogeneous FPGA architectures: application specific exploration and optimization. Springer Science & Business Media, 2012, pp. 186.
3. *Singh A., Parthasarathy G., Marek-Sadowska M.* Interconnect resource-aware placement for hierarchical FPGAs, *IEEE/ACM International Conference on Computer Aided Design. ICCAD 2001. IEEE/ACM Digest of Technical Papers (Cat. No. 01CH37281)*. IEEE, 2001, pp. 132-136.
4. *Boutros A., Betz V.* FPGA architecture: Principles and progression, *IEEE Circuits and Systems Magazine*, 2021, Vol. 21, No. 2, pp. 4-29.
5. PLIS emkost'yu 21 502 ventiley 5510TS068 [FPGA with a capacity of 21,502 gates 5510TS068]. JSC Mikron. Available at: https://mikron.ru/products/high-rel-ic/programmiruemaya-logika-fpga/fpga/product/5510ts068/ (accessed 29 January 2022).
6. *Petelin O., Betz V.* Wotan: evaluating FPGA architecture routability without benchmarks, *ACM Trans. Reconfigurable Technol. Syst*., 2018, Vol. 11, No. 2, pp. 1-23.
7. *Frolova P.I., Chochaev R.Zh., Ivanova G.A., Gavrilov S.V.* Algoritm razmeshcheniya s optimizatsiey bystrodeystviya na osnove matrits zaderzhek dlya rekonfiguriruemykh sistem na kristalle [Placement algorithm with performance optimization based on delay matrices for reconfigurable systems on a chip], *Problemy razrabotki perspektivnykh mikro- i nanoelektronnykh sistem (MES)* [Problems in the development of advanced micro- and nanoelectronic systems (MES)], 2020, Issue 1, pp. 2-7.
8. *Frolova P.I., Khvatov V.M., Chochaev R.Zh.* Sravnitel'nyy analiz metodov klasterizatsii i razmeshcheniya skhem dlya rekonfiguriruemykh sistem na kristalle [Comparative analysis of clustering methods and circuit placement for reconfigurable systems on a chip], *Problemy razrabotki perspektivnykh mikro- i nanoelektronnykh sistem (MES)* [Problems of development of advanced micro- and nanoelectronic systems (MES)], 2022, Issue 4, pp. 63-70.

9. *Kureichik V.V., Zaporozhets D.Y., Zaruba D.V.* Partitioning of VLSI fragments based on the model of glowworm's behavior, *Proceedings of the 19th International Conference on Soft Computing and Measurements, SCM 2016,* 2016, pp. 268-272.
10. *Li W. et al.* UTPlaceF 3.0: A parallelization framework for modern FPGA global placement, *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2017, pp. 922-928.
11. *Chen G., et al.* RippleFPGA: Routability-driven simultaneous packing and placement for modern FPGAs, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2017, Vol. 37, No. 10, pp. 2022-2035.
12. *Dhar S., et al.* Detailed placement for modern FPGAs using 2D dynamic programming, *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD).* IEEE, 2016, pp. 1-8.
13. *Nikolić S., Zgheib G., Ienne P.* Detailed Placement for Dedicated LUT-Level FPGA Interconnect, *ACM Transactions on Reconfigurable Technology and Systems*, 2022, Vol. 15, No. 4, pp. 1-33.
14. *Gavrilov S.V., Zheleznikov D.A., Chochaev R.Zh.* Razrabotka i sravnitel'nyy analiz metodov resheniya zadachi razmeshcheniya dlya rekonfiguriruemykh sistem na kristalle [Development and comparative analysis of methods for solving the placement problem for reconfigurable systems on a chip], *Izvestiya vysshikh uchebnykh zavedeniy. Elektronika* [News of higher educational institutions. Electronics], 2020, Vol. 25, No. 1, pp. 48-57.
15. *Fan D.K., Shi P.* Improvement of Dijkstra's algorithm and its application in route planning, *2010 seventh international conference on fuzzy systems and knowledge discovery*. IEEE. 2010, Vol. 4, pp. 1901-1904.
16. *Dijkstra E.W.* A note on two problems in connexion with graphs, *Edsger Wybe Dijkstra: His Life, Work, and Legacy,* 2022, pp. 287-290.
17. *Bryan D.* The ISCAS’85 benchmark circuits and netlist format, *North Carolina State University,* 1985, Vol. 25, pp. 39.
18. *Brglez F., Bryan D., Kozminski K.* Combinational profiles of sequential benchmark circuits, *1989 IEEE International Symposium on Circuits and Systems (ISCAS).* IEEE, 1989, pp. 1929-1934.
19. 8080 Compatible CPU. Available at: https://opencores.org/projects/cpu8080.
20. The OpenCores VGA/LCD Controller. Available at: https://opencores.org/projects/vga_lcd.

Статью рекомендовал к опубликованию д.т.н., профессор В.В. Курейчик.

Шокарев Дмитрий Борисович – Институт проблем проектирования в микроэлектронике РАН; e-mail: shokarev_d@ippm.ru; Москва, Зеленоград, Россия; тел.: +74997299890; отдел систем автоматизированного проектирования интегральных схем; инженер-исследователь.

Чочаев Рустам Жамболатович – e-mail: chochaev_r@ippm.ru; тел.: +74997299890; отдел систем автоматизированного проектирования интегральных схем; инженер-исследователь.

Щелоков Альберт Николаевич – e-mail: schan@ippm.ru; тел.: +74997299845; отдел проектирования микроэлектронных компонентов для нанотехнологий; к.ф.-м.н.; доцент; помощник директора.

Гаврилов Сергей Витальевич – e-mail: sergey_g@ippm.ru; тел.: +74997299845; д.т.н.; директор; профессор Института интегральной электроники имени академика К.А. Валиева Национального исследовательского университета «МИЭТ».

Shokarev Dmitry Borisovich – The Institute for Design Problems in Microelectronics; e-mail: shokarev_d@ippm.ru; Moscow, Zelenograd, Ruusia; phone: +74997299890; the department of computer-aided design of integrated circuits; research engineer.

Chochaev Rustam Zhambolatovich – e-mail: chochaev_r@ippm.ru; phone: +74997299890; the department of computer-aided design of integrated circuits; research engineer.

Schelokov Albert Nikolaevich – e-mail: schan@ippm.ru; phone: +74997299845; the department of design of microelectronic components for nanotechnology; cand. of phys. and math. sc.; associate professor; director’s assistant.

Gavrilov Sergey Vitalievich – e-mail: sergey_g@ippm.ru; phone: +74997299845; dr. of eng. sc.; director; professor at the Institute of Integrated Electronics, National Research University of Electronic Technology (MIET).