УДК 004.02:004.415.23:621.3.049.771.14 DOI: 10.31114/2078-7707-2022-4-63-70

Сравнительный анализ методов кластеризации и размещения схем для реконфигурируемых систем на кристалле

П.И. Фролова, В.М. Хватов, Р.Ж. Чочаев

Институт проблем проектирования в микроэлектронике РАН (ИППМ РАН), г. Москва

frolova_p@ippm.ru, khvatov_v@ippm.ru, chochaev_r@ippm.ru

***Аннотация* — Реконфигурируемые системы на кристалле (РСнК) занимают все большую долю на рынке микро- и наноэлектроники. Наличие программируемой части в совокупности с жесткими сложными функциональными (СФ) блоками на одном кристалле делает их универсальными и пригодными для выполнения разного рода задач. При проектировании интегральных схем (ИС) на основе РСнК остро стоит вопрос быстродействия схем. Оно зависит как и от архитектуры целевой реконфигурируемой системы, так и от результатов каждого этапа маршрута проектирования. Основополагающим фактором, влияющим на скорость распространения сигнала между логическими элементами, является их расположение относительно друг друга. Это определяется на этапе кластеризации и размещения элементов разрабатываемой схемы на базе РСнК. В данной статье эти два этапа рассматриваются совместно, т.к. они неразрывно связаны друг с другом и решают общую задачу распределения логических элементов схемы по группам логических элементов РСнК. Для выполнения кластеризации и размещения выбраны методы, использующие программный модуль KaHyPar, последовательный многоуровневый алгоритм размещения (ПМАР) и стандартный плоский алгоритм размещения (СПАР). Результатом работы является сравнительный анализ представленных методов. В качестве оценки используются такие параметры как количество трассировочных элементов в цепях, количество цепей с определенным числом трассировочных элементов, а также трассируемость схем. На основе полученных результатов показаны преимущества и недостатки представленных методов.**

***Ключевые слова* — кластеризация, размещение, имитация отжига, KaHyPar, СнК, САПР.**

I. Введение

В маршруте автоматизированного проектирования интегральных схем (ИС) на основе реконфигурируемых систем на кристалле (РСнК) и программируемых логических интегральных схемах (ПЛИС) все этапы неразрывно связаны друг с другом. Нельзя обособленно рассматривать конкретный этап маршрута, так как его влияние на другие этапы может быть значительным. Примером этого является взаимосвязь этапов кластеризации и размещения. Кластеризация представляет собой деление списка соединений разрабатываемой схемы на группы логических элементов (ГЛЭ). Размещение элементов выполняется на основе полученного разбиения и представляет собой перемещение ГЛЭ, перемещение ЛЭ внутри этих групп и легализацию их позиций. Эти два этапа могут быть выполнены как по отдельности, так и совокупности друг с другом. От результата кластеризации в дальнейшем зависит качество размещения, и, как следствие, трассируемость разрабатываемой схемы [1].

В данной работе исследована зависимость качества размещения и трассировки от методов кластеризации и последующего этапа размещения. Проведено сравнение двух алгоритмов кластеризации, и двух алгоритмов размещения.

Примером существующих методов кластеризации являются hMetis [2], PaToH [3], Zoltan [4], Mondriaan [5]. Они широко используются в индустрии, и зачастую являются основой для более совершенных модификаций. В свою очередь, алгоритмы размещения традиционно делятся на 3 класса: аналитические, дихотомические и эвристические. Примером первых служат такие алгоритмы как SImPL[6] и HeAP [7]. Ко второму классу относятся алгоритмы [8-9]. А среди представителей последнего класса можно выделить следующие алгоритмы на основе метода имитации отжига [10] и генетические алгоритмы [11].

Первый алгоритм кластеризации, который будет использован в данной работе, выбран из программного модуля KaHyPar (Karlsruhe Hypergraph Partitioning) [12]. Благодаря тому, что программное обеспечение KaHyPar находится в свободном доступе, данный модуль можно легко использовать в различных исследованиях. Подход к разбиению графа, использующийся в KaHyPar, отличается от других тем, что он использует комбинации сразу нескольких эффективных алгоритмов, а также имеет более широкий спектр параметров для тонкой настройки режимов работы. Он будет подробно рассмотрен в главе, посвященной описанию используемых методов. В качестве альтернативного решения выбран алгоритм iRAC [13-14], который представляет собой двухэтапный алгоритм кластеризации специализированный под использование для различных архитектур ПЛИС. Его главной целью является уменьшение количества внешних цепей между кластерами. Оценка перегруженности в нем

выполняется с помощью правила Рента. iRAC реализован в исследуемом маршруте и используется совместно с двумя методами размещения.

Среди множества алгоритмов и методов размещения в данной работе выбраны последовательный многоуровневый алгоритм размещения (ПМАР) [15] и стандартный плоский алгоритм размещения (СПАР) [15]. Оба алгоритма размещения используют эвристический подход и состоят из нескольких внутренних этапов. Эвристикой для них являются разного рода оптимизации, реализованные с помощью метода имитации отжига. Алгоритмы отличаются конкретными задачами, решаемыми в рамках имеющихся этапов, а также используемыми иерархическими моделями.

Статья организована следующим образом. В главе 2 рассмотрена архитектура РСнК и связанные с ней особенности размещения; третья глава посвящена используемым алгоритмам кластеризации и размещения разрабатываемой схемы. Экспериментальные результаты представлены в главе 4 с последующими выводами.

II. Архитектура РСнК

Исследование методов кластеризации и размещения проводилось на основе РСнК 5510ТС054, разработанной в АО “НИИМЭ” и произведенной на ПАО “Микрон” с использованием технологии КМОП КНИ 180 нм. Рассмотренная РСнК является косвенным аналогом ПЛИС ProAsicPlus (APA1000) компании Microchip (Actel) и состоит из логических элементов (ЛЭ), ячеек ввода-вывода (ЯВВ) и набора макроблоков, к которым относятся блоки памяти, умножители и блоки фазовой автоподстройки частоты [16].

Логический элемент, по аналогии с ProAsicPlus, представляет собой универсальную ячейку с тремя входами, состоящую из проходных ключей и набора логических вентилей [17-18]. Он может быть сконфигурирован либо как элемент комбинационной логики, либо как триггер со входами для тактового сигнала, данных и сброса.

ЛЭ на РСнК объединены в группы логических элементов (ГЛЭ). Каждая группа состоит из 256 ЛЭ, расположенных в виде массива из 16 строк и 16 столбцов ЛЭ. В состав РСнК входит 100 ГЛЭ, распределенных по кристаллу в виде массива из 10 строк и 10 столбцов. По периметру массива ГЛЭ расположены ЯВВ, сверху и снизу – макроблоки, связанные как с ЯВВ, так и с реконфигурируемой логикой. Схематичное расположение ЛЭ, ЯВВ и макроблоков левой верхней части РСнК изображено на рис. 1.

Соединения ЛЭ образуют трассировочную архитектуру, названную в зарубежной литературе – “море ячеек” (от англ. “sea-of-gates”) [19]. Она включает в себя два основных типа связей – локальные и глобальные. С помощью локальных связей каждый ЛЭ соединен с ближайшими соседними элементами по вертикали и горизонтали. Глобальные связи проходят через весь кристалл, через все строки и столбцы ГЛЭ, и позволяют соединить ЛЭ на любом расстоянии друг от друга. Кроме этого, в рассматриваемую архитектуру входят дополнительные связи, позволяющие соединить ЛЭ со 2-ым, 3-им и 4-ым соседним ЛЭ по вертикали и горизонтали.

Рис. 1. Архитектура ПЛИС

III. Используемые методы кластеризации и размещения

В данной работе рассматриваются три метода совместного выполнения кластеризации и размещения ИС на основе РСнК: метод на основе ПО KaHyPar, метод на основе последовательного многоуровневого алгоритма размещения (ПМАР) и на основе стандартного плоского алгоритма размещения (СПАР). Все используемые методы включают в себя кластеризацию ЛЭ по ГЛЭ, глобальное размещение ГЛЭ на кристалле и детальное размещение ЛЭ внутри группы. Однако от выбранного метода зависит алгоритм кластеризации и наличие дополнительных этапов при выполнении размещения.

Последовательность выполнения этапов в каждом методе показана на рис. 2. В первом методе используется кластеризация с помощью модуля KaHyPar, иерархическое глобальное размещение в совокупности с моделированием отжига и детальное размещение ЛЭ внутри групп. Во втором методе, использующем ПМАР, кластеризация выполняется алгоритмом iRAC, а размещение аналогично первому методу. Последний метод совместной кластеризации и размещения использует СПАР и включает в себя кластеризацию алгоритмом iRAC, а также размещение с дополнительным этапом - плоским размещением. В отличие от двух рассмотренных методов, где перемещение ЛЭ между группами возможно только на стадии кластеризации, данный метод позволяет

выполнить такую процедуру после получение оптимального результата глобального размещения, тем самым учитывая особенности архитектуры целевой РСнК.

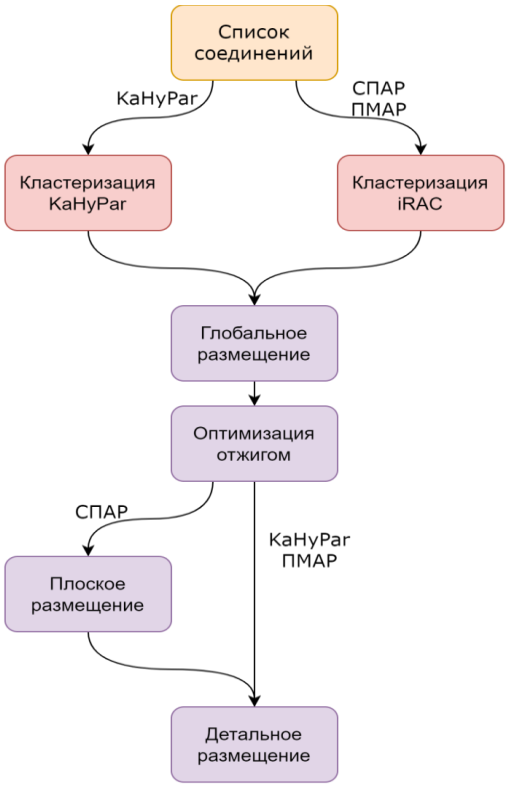

Рис. 2. Схема этапов алгоритмов

A. Метод с использованием ПО KaHyPar

KaHyPar – это программный модуль, выполняющий разбиение гиперграфов. Он разбивает множество вершин на k непересекающихся блоков ограниченного размера (не более чем в $1 + \varepsilon$ раз больше среднего размера блока), при этом минимизируя целевую функцию, определенную для каждого ребра, соединяющего блоки.

Данный процесс в KaHyPar состоит из трех этапов. На первом этапе гиперграф разделяется на гиперграфы меньшего размера, для каждого из которых на втором этапе выполняется начальное разбиение. Далее, на третьем этапе, происходит оптимизация результатов алгоритмами локального поиска. Для разбиения гиперграфа на выбор пользователя предлагается два алгоритма – алгоритм разделения на основе прямого k-пути [20] и алгоритм рекурсивного деления пополам [21]. Для оптимизации результата разбиения доступны две целевые функции – на основе стандартного разреза графа и на основе разреза с учетом связности блоков (λ-1 метрика). В первом случае минимизируется сумма весов ребер, соединяющих более одного блока, во-втором сумма весов дополнительно умножается на количество блоков (λ-1), с которыми соединено рассматриваемое ребро.

KaHyPar имеет различные конфигурации запуска, которые позволяют запустить необходимый алгоритм разбиения гиперграфа и выбрать требуемую целевую функцию, оптимально подходящую под конкретную задачу. В рамках работы для всех исследуемых методов был использованы одинаковые параметры ПО: представленный разработчиками файл конфигурации, использующий алгоритм разделения на основе прямого k-пути и λ-1 метрику; а также параметр, отвечающий за количество создаваемых кластеров, заданный в соответствии с количеством ГЛЭ рассматриваемой РСнК.

B. Метод на основе алгоритма иерархического размещения ПМАР

ПМАР является иерархическим алгоритмом и включает в себя два основных этапа размещения и один дополнительный этап кластеризации.

На этапе кластеризации схема подготавливается к первому этапу размещения: глобальному, для этого она разбивается на необходимое количество групп логических элементов (ГЛЭ). Используемый метод кластеризации представляет из себя модифицированный алгоритм iRAC. Подробное описание данной реализации алгоритма кластеризации приведено в статье [15].

Алгоритм iRAC — это восходящий алгоритм кластеризации, направленный на повышение трассируемости ПЛИС. В работе [13] показывается, что своевременный учет архитектурных особенностей межсоединений позволяет сократить длины цепей, и, как следствие, уменьшить площадь и потребляемую мощность. С помощью правила Рента [22-23] в алгоритме оценивается связность кластеров для более плотной упаковки и кластерной однородности. Экспонента Рента позволяет оценить сложность межсоединений архитектуры и зависит от количества элементов в схеме, количества внешних терминалов схемы, и количества соединений, приходящихся на один элемент в схеме. Значение экспоненты варьируется от 0 до 1. К целям кластеризации относятся минимизация количества внешних цепей, которые необходимо трассировать, и согласование параметра Рента сформированных кластеров с параметром базовой архитектуры ПЛИС.

Для каждого логического элемента вычисляется показатель связности и формируется отсортированный список. Далее формируются кластеры. Стоит отметить, что особая роль уделяется уменьшению количества внешних связей формируемого кластера, что положительно сказывается на итоговой длине цепей межсоединений. Кластер формируется из ЛЭ с наибольшим значением коэффициента притяжения, который зависит от веса цепи и количества терминалов цепи внутри кластера. Дополнительно, количество внешних терминалов в кластере ограничивается посредством правила Рента.

Как только кластеры получены, начинается этап глобального размещения. Для генерации начального используется силовой алгоритм. Когда для всех групп элементов определены конкретные позиции

размещения заканчивается. Далее, полученное размещение оптимизируется с помощью модифицированного метода имитации отжига.

После глобального следует этап детального размещения. Аналогично глобальному оно состоит из двух шагов – генерации начального размещения и его оптимизации методом отжига. В качестве целевой функции используется оценка длин межсоединений.

C. Метод на основе алгоритма плоского размещения СПАР

Следующий используемый в работе метод основан на алгоритме плоского размещения и описан в работе [15]. СПАР состоит из двух этапов начального размещения и его последующей оптимизации методом имитации отжига.

Далее в алгоритме реализуется стадия плоского размещения. На ней убирается иерархия схемы, что позволяет более эффективно учитывать внешние связи кластеров и размещать ЛЭ из разных групп ближе друг к другу. Также появляется возможность задействовать вместо длинных и ограниченных по количеству глобальных связей более быстрые короткие локальные связи.

IV. Экспериментальные результаты

В данной работе был выбран набор тестовых схем разного размера (от 5-27 тыс. ЛЭ), для которых предварительно была проведена отладка [24], логический синтез [25] и технологическое отображение [26] в базис РСнК. Алгоритмы сравнивались по следующим критериям: трассируемость, количество трассировочных элементов в цепи, количество цепей с конкретным числом трассировочных элементов. Отдельно выделена статистика по минимальному, максимальному и среднему количеству трассировочных элементов в цепи.

В рамках работы введем понятие «длина цепи», которое будет выражаться числом трассировочных элементов в этой цепи. Такое допущение позволяет ввести сегментированную и однородную структуру трассировочных ресурсов. Цепью в данном случае называется соединение между двумя ЛЭ, то есть от выходного контакта элемента-источника до входного контакта элемента-приемника.

В табл. 1 сформированы результаты для 4-х схем, размеры которых не превышают 10 тыс. логических ячеек. Длина самой длинной цепи в самом худшем случае для данного поднабора схем не превышает 300 элементов. Вследствие большого количества элементов в цепи для упрощения визуализации результатов полученный диапазон значений разбит на равные интервалы по 25 значений в каждом. Для каждого диапазона приводится число цепей, длина которых находится в рамках указанного диапазона.

Таблица 1

Распределение цепей с находящимся в интервале числом трассировочных элементов

Схема	Количество цепей											
	ac97_top			ae18_core			aes_ip			sha256		
Интервал	KaHyPar	ПМАР	СПАР	KaHyPar	ПМАР	СПАР	KaHyPar	ПМАР	СПАР	KaHyPar	ПМАР	СПАР
1-25	13744	10885	14468	12691	9879	12955	13019	10961	13699	16747	14290	18197
26-50	2481	2526	2541	2194	2723	2364	2586	2976	2845	2248	2692	2208
51-75	897	1642	512	653	1417	475	1108	1931	1068	952	1459	780
76-100	296	1085	91	260	765	164	756	1030	554	469	904	392
101-125	117	636	13	120	471	67	536	763	204	480	730	175
126-150	53	334		81	316	33	316	434	109	432	601	122
151-175	37	179		60	175	1	194	253	130	246	486	114
176-200		81			114		83	144	85	154	347	24
201-225		33			91		81	123	27	108	225	2
226-250		18			49		34	61		80	116	
251-275		93			37		8	26		36	112	
276-300		113			22			19		62	52	
Всего	17625	17625	17625	16059	16059	16059	18721	18721	18721	22014	22014	22014

По приведенным в табл. 1 данным видно, что метод, использующий алгоритм СПАР, позволяет на интервале длиной 1-25 получить коротких цепей на 5% больше, чем метод на основе KaHyPar, и на 25% больше, чем метод на основе ПМАР. Кластеризация методом KaHyPar несколько уступает ему на самых коротких цепях, но компенсирует это отставание в диапазоне 50-100 элементов в цепи.

У всех исследуемых методов значительное число цепей находится в диапазоне до 25 элементов в длину. На него приходится в среднем 76%, 62% и 80% элементов цепи соответственно используемому методу, что закрепляет преимущество плоского метода размещения.

Разброс в длине цепей для алгоритмов колеблется в пределах одного-трех интервалов и может достигать 75 элементов. Метод на основе алгоритма ПМАР существенно проигрывает им обоим и вынуждает использовать излишне длинные цепи трассировки (до 300 элементов). По сравнению с методом на основе СПАР разрыв может составлять пять интервалов, что говорит о разнице в не менее чем 100 элементов. Для всех схем из таблицы 1 во всех режимах обеспечивалась 100% трассируемость.

В табл. 2 отображены результаты для трех схем большего размера (15-27 тыс. ЛЭ). Максимальная длина цепей в них возрастает до 325 элементов, за счет чего формируется дополнительный интервал. Выявленная тенденция сохраняется, при этом алгоритму KaHyPar удалось сократить разрыв значений максимальных длин цепей для данного набора схем, что оказало влияние на усредненное значение количества элементов в цепях. А между алгоритмами СПАР и ПМАР разница в длине возросла до 7 интервалов и составила 50%.

При работе со второй частью исследуемого набора схем у каждого метода ухудшилась трассируемость – как минимум 1 из 3 схем не была разведена на 100%. Это хорошо видно из общего количества использованных при трассировке элементов, который также указан в табл. 2. При использовании метода на основе KaHyPar не была разведена только схема fpu, при размещении методом на основе иерархического размещения не была разведена ни одна схема, при использовании метода на основе плоского размещения не разведена схема shavite_top. Число неразведенных цепей невелико, из ~20000 цепей не удалось развести от 1 до 5 цепей, что составляет ~0,02% от их общего числа. Трассировка проводилась алгоритмом Pathfinder.

Таблица 2

Распределение цепей с находящимся в интервале числом трассировочных элементов

Схема	Количество цепей								
	fpu			riscv_core			shavite_top		
Интервал	KaHyPar	ПМАР	СПАР	KaHyPar	ПМАР	СПАР	KaHyPar	ПМАР	СПАР
1-25	30092	24900	33139	43142	37432	45414	38718	28803	40767
26-50	3871	4715	3854	6058	7683	6040	8365	9499	8556
51-75	1924	2556	1080	3615	4818	3081	4862	6203	3946
76-100	1333	1591	157	2349	2944	1596	2905	4175	2083
101-125	593	1092	31	1407	1805	855	1541	2779	1333
126-150	338	850	3	812	1213	548	745	1956	754
151-175	91	744		511	788	350	615	1281	472
176-200	20	580		406	450	348	498	802	221
201-225		407		256	309	281	332	719	268
226-250		270		295	252	203	231	677	264
251-275		215		201	220	217	164	745	314
276-300		323		246	798	365	142	1404	134
301-325		1		1	9	1		51	
Всего	38262	38244	38264	59299	58796	59299	59118	59094	59112

От количества элементов в цепях зависит быстродействие схемы, поэтому рассмотрим подробнее полученные результаты. На рис. 3-5 приведены данные о среднем, минимальном и максимальном количестве элементов в цепях для разных алгоритмов.

Как видно из рис. 3 результаты работы алгоритмов не зависят от выбора схемы. В проведенном эксперименте явно выделяется следующая тенденция: алгоритм СПАР использует в среднем меньше всего элементов в отдельно взятой цепи, далее за ним с небольшим отставанием идет алгоритм KaHyPar и замыкает список алгоритм ПМАР, причем с существенным отставанием, в отдельных случаях достигающим 40-45%.

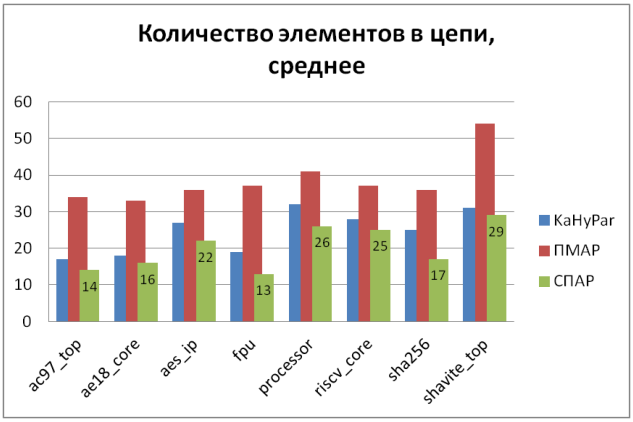

Рис. 3. Диаграмма количества элементов в цепях

Полученная картина несколько отличается для максимальных значений (рис. 4). Для ряда схем из экспериментального набора (ae18_core, processor,

riscv_core, shavite_top) значения, полученные при оценке алгоритмов СПАР и KaHyPar, практически одинаковы и достигают значения в 300+ элементов.

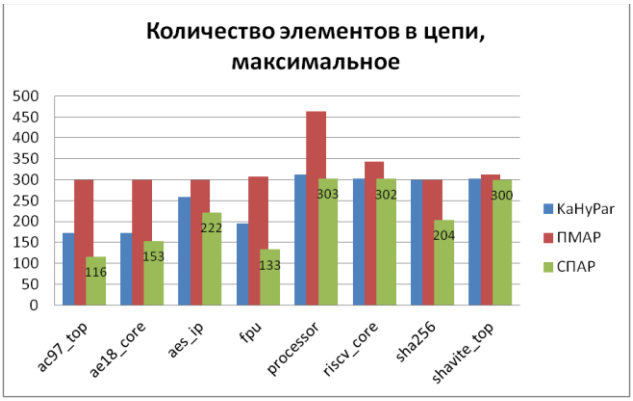

Рис. 4. Диаграмма количества элементов в цепях

Также стоит отдельно обратить внимание на схему shavite_top. От других ее отличает то, что все три алгоритма справились с ней приблизительно одинаково с точки зрения максимального количества трассировочных элементов в отдельно взятой цепи.

В целом же, алгоритм СПАР сохранил свою лидирующую позицию по результатам данного сравнения.

Результаты сравнения длин минимальных цепей показывают, что этот параметр больше зависит от трассировочной архитектуры ПЛИС, чем от используемых алгоритмов, наблюдаемая разница в 1 элемент для некоторых схем не окажет существенного влияния на работе схем.

Как видно из рис. 3-5 алгоритм СПАР демонстрирует наилучшие показатели для всех трех измерений, то есть он стабильно позволяет алгоритму трассировки использовать более короткие цепи, чем у его конкурентов.

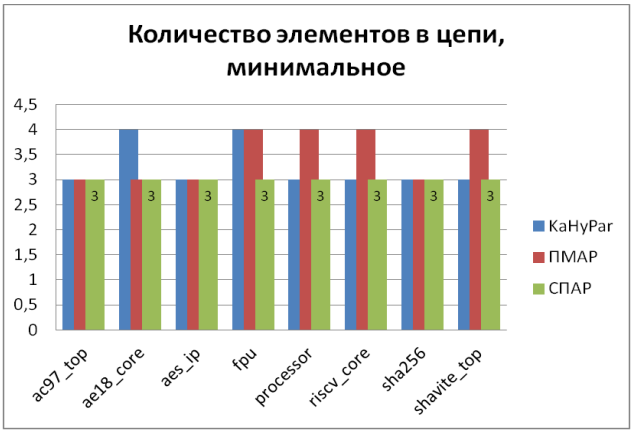

Рис. 5. Диаграмма количества элементов в цепях

V. Выводы

В рамках данной работы было проведено исследование трех методов кластеризации с последующим размещением пользовательских схем на реконфигурируемой системе на кристалле. Рассмотренные методы основаны на ПО KaHyPar и алгоритмах ПМАР и СПАР, соответственно.

Данные методы отличаются подходами к кластеризации и использованием информации об архитектуре РСнК при размещении на ней логических элементов. Для сравнительной оценки алгоритмов были выбраны следующие параметры: трассируемость, количество трассировочных элементов в цепи, количество цепей с конкретным числом трассировочных элементов и статистика по максимальному, минимальному и среднему значению по количеству элементов в цепи.

Исходя из полученных результатов, наилучшие результаты показал метод на основе алгоритм СПАР. Он позволяет получить более короткие соединения между ЛЭ, чем два других метода, что косвенно повышает быстродействие разрабатываемой схемы на основе рассмотренной РСнК.

Литература

[1] Заплетина М.А., Железников Д.А., Гаврилов С.В. Иерархический подход к трассировке реконфигурируемой системы на кристалле островного типа // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2020. Выпуск 3. С. 16-21. doi:10.31114/2078-7707-2020-3-16-21.

[2] Miettinen P., Honkala M., Roos J. Using METIS and hMETIS Algorithms in Circuit Partitioning // Report CT-49, Circuit Theory Laboratory, Helsinki University of Technology, 2006.

[3] Çatalyürek Ü., Aykanat C. PaToH (Partitioning Tool for Hypergraphs) // Encyclopedia of Parallel Computing, 2020. Springer US, Boston, MA, pp. 1479-1487. https://doi.org/10.1007/978-0-387-09766-4_93

[4] Devine K. D. , Boman E.G. , Riesen L.A. , Catalyurek U.V., Chevalier C. Getting started with zoltan: A short tutorial. // In Combinatorial Scientific Computing №09061 in Dagstuhl Seminar Proceedings, 2009, p. 10

[5] Vastenhouw B., Bisseling R. A Two-Dimensional Data Distribution Method For Parallel Sparse Matrix-Vector Multiplication // SIAM Review, 47, No. 1, 2005 pp. 67-95. doi: 10.1137/S0036144502409019.

[6] Kim M. C., Lee D., Markov I. SimPL: An effective placement algorithm // IEEE TCAD, vol. 31, no. 1, 2012, pp. 50-60.

[7] Gort M., Anderson J. H. Analytical placement for heterogeneous FPGAs // 22nd International Conference on Field Programmable Logic and Applications (FPL), 2012, pp. 143-150. doi: 10.1109/FPL.2012.6339278.

[8] Hutton M., Adibsamii K. Leaver. A. Adaptive delay estimation for partitioning-driven PLD placement // IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 11, no. 1, pp. 60-63, Feb. 2003. doi: 10.1109/TVLSI.2002.808424.

[9] Rose J., Snelgrove W., Vranesic Z. ALTOR: An automatic standard cell layout program // Proceedings of the Canadian Conference on VLSI, 1985, pp. 169 - 173.

[10] 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.

[11] Лебедев Б. К., Степаненко С. А. Генетический алгоритм размещения, управляемый временными ограничениями // Известия ЮФУ. Технические науки. 2007. №1.
[12] Schlag S., Heuer T., Gottesbüren L., Akhremtsev Y., Schulz C., Sanders P. High-Quality Hypergraph Partitioning // ACM J. Exp. Algorithmics Just Accepted, Association for Computing Machinery, New York, 2022. doi: 10.1145/3529090.
[13] Singh A., Parthasarathy G., Marek-Sadowska, M. Efficient circuit clustering for area and power reduction in FPGAs // ACM Trans. Design Autom. Electr. Syst. vol. 7, №4, pp. 643-663. doi:10.1145/605440.605448.
[14] 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. № 1. С. 199-204. doi: 10.31114/2078-7707-2018-1-199-204.
[15] 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., doi: 10.24151/1561-5405-2020-25-1-48-57.
[16] Лот 7. ОКР «Разработка и освоение производства радиационно-стойкой отказоустойчивой ПЛИС емкостью не менее 250 тыс. логических вентилей со встроенными блоками PLL и умножителями», шифр «Алмаз-14». URL: https://zakupki.gov.ru/epz/order/notice/ok44/view/documents.html?regNumber=0173100009515000200 (дата обращения: 26.08.2022)
[17] ProASIC3 FPGA Fabric User's Guide URL: https://www.microsemi.com/document-portal/doc_download/130708-proasic-sup-u-plus-u-sup-flash-family-fpgas-datasheet (дата обращения: 26.08.2022)
[18] Карпов, С. Flash-семейства ПЛИС "Актел" / С. Карпов // Компоненты и технологии. – 2007. – № 11(76). – С. 56-62.
[19] Vassiliadis S., Soudris D. Fine- and Coarse-Grain Reconfigurable Computing. Springer, 2007. 381 c.
[20] Karypis G., Kumar V. Parallel multilevel k-way partitioning scheme for irregular graphs // In Proceedings of the 1996 ACM/IEEE conference on Supercomputing (Supercomputing '96), 1996, IEEE Computer Society, USA, p. 21.
[21] Schlag S., et al. k-way Hypergraph Partitioning via n-Level Recursive Bisection URL: https://arxiv.org/abs/1511.03137 (дата обращения: 26.08.2022)
[22] Donath W. Placement and average interconnection lengths of computer logic // IEEE Transactions on Circuits and Systems, vol. 26, no. 4, pp. 272-277, April 1979, doi: 10.1109/TCS.1979.1084635.
[23] Landman B. S., Russo R. L. On a Pin Versus Block Relationship For Partitions of Logic Graphs // IEEE Transactions on Computers, vol. C-20, no. 12, pp. 1469-1479, Dec. 1971, doi: 10.1109/T-C.1971.223159.
[24] Иванников, А. Д. Формализация выбора отладочных тестов при проектировании цифровых микроэлектронных систем на основе проверки выполнения требуемых функций / А. Д. Иванников, А. Л. Стемпковский // Известия высших учебных заведений. Электроника. – 2020. – Т. 25. – № 4. – С. 310-319. – doi: 10.24151/1561-5405-2020-25-4-310-319.
[25] D. Shah, E. Hung, C. Wolf, S. Bazanski, D. Gisselquist and M. Milanovic, "Yosys+nextpnr: An Open Source Framework from Verilog to Bitstream for Commercial FPGAs," 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2019, pp. 1-4, doi: 10.1109/FCCM.2019.00010.
[26] Тиунов И.В., Липатов И.А., Железников Д.А. Разработка методов архитектурно-ориентированного ресинтеза в маршруте автоматизированного проектирования ПЛИС. Проблемы разработки перспективных микро- и наноэлектронных систем. 2018. Выпуск I. С. 69-74. doi:10.31114/2078-7707-2018-1-69-74.

Comparative Analysis of Clustering and Placement Methods for Reconfigurable System-on-Chips

P.I. Frolova, V.M. Khvatov, R.Zh. Chochaev

Institute for Design Problems in Microelectronics of RAS, Moscow

frolova_p@ippm.ru, khvatov_v@ippm.ru, chochaev_r@ippm.ru

***Abstract* — The share of reconfigurable systems on a chip (RSoC) in the micro- and nanoelectronics market is increasing. The programmable part in combination with hard IP-cores on a single chip makes them versatile and suitable for various tasks. When designing integrated circuits (ICs) based on RSoC, the issue of their performance is acute. It depends both on the architecture of the target reconfigured circuit and on the results of each stage of the design flow. The key factor influencing the signal propagation time between logic elements (LE) is their relative position to each other. This is determined during clustering and placement of LEs of the developed circuit based on the RSoC. In this article, these two stages are considered together, since they are inextricably linked with each other and solve the general problem of distributing logical elements into RSoC logic array blocks (LABs). Several methods have been chosen to perform clustering and placement. The first method uses the KaHyPar framework for clustering. The second method represents the global placement of LABs and sequential detailed placement of logic elements within LABs. In the third method, after the global placement step, the rearrangement of logical elements without LAB boundaries is performed. The result of the work is a comparative analysis of the presented methods. To evaluate the methods, parameters such as the number of routing elements in the net, the number of nets with a certain number of routing elements, and the routability are used. The article also shows the advantages and disadvantages of**

the presented methods, identified on the basis of the obtained results.

***Keywords* — clustering, placement, simulated annealing, KaHyPar, SoC, CAD.**

REFERENCES

[1] Zapletina M.A., Zheleznikov D.A., Gavrilov S.V. The Hierarchical Approach to Island Style Reconfigurable System-on-a-chip Routing // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2020. Issue 3. P. 16-21. doi:10.31114/2078-7707-2020-3-16-21.

[2] Miettinen P., Honkala M., Roos J. Using METIS and hMETIS Algorithms in Circuit Partitioning // Report CT-49, Circuit Theory Laboratory, Helsinki University of Technology, 2006.

[3] Çatalyürek Ü., Aykanat C. PaToH (Partitioning Tool for Hypergraphs) // Encyclopedia of Parallel Computing, 2020. Springer US, Boston, MA, pp. 1479-1487. https://doi.org/10.1007/978-0-387-09766-4_93

[4] Devine K. D. , Boman E.G. , Riesen L.A. , Catalyurek U.V., Chevalier C. Getting started with zoltan: A short tutorial. // In Combinatorial Scientific Computing №09061 in Dagstuhl Seminar Proceedings, 2009, p. 10

[5] Vastenhouw B., Bisseling R. A Two-Dimensional Data Distribution Method For Parallel Sparse Matrix-Vector Multiplication // SIAM Review, 47, No. 1, 2005 pp. 67-95. doi: 10.1137/S0036144502409019.

[6] Kim M. C., Lee D., Markov I. SimPL: An effective placement algorithm // IEEE TCAD, vol. 31, no. 1, 2012, pp. 50-60.

[7] Gort M., Anderson J. H. Analytical placement for heterogeneous FPGAs // 22nd International Conference on Field Programmable Logic and Applications (FPL), 2012, pp. 143-150. doi: 10.1109/FPL.2012.6339278.

[8] Hutton M., Adibsamii K. Leaver. A. Adaptive delay estimation for partitioning-driven PLD placement // IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 11, no. 1, pp. 60-63, Feb. 2003. doi: 10.1109/TVLSI.2002.808424.

[9] Rose J., Snelgrove W., Vranesic Z. ALTOR: An automatic standard cell layout program // Proceedings of the Canadian Conference on VLSI, 1985, pp. 169 - 173.

[10] 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.

[11] Lebedev B. K., Stepanenko S. A. Geneticheskiy algoritm razmeshcheniya, upravlyayemyy vremennymi ogranicheniyami (Genetic placement algorithm driven by time constraints) // Izvestiya YUFU. Tekhnicheskiye nauki. 2007. №1.

[12] Schlag S., Heuer T., Gottesbüren L., Akhremtsev Y., Schulz C., Sanders P. High-Quality Hypergraph Partitioning // ACM J. Exp. Algorithmics Just Accepted, Association for Computing Machinery, New York, 2022. doi: 10.1145/3529090.

[13] Singh A., Parthasarathy G., Marek-Sadowska, M. Efficient circuit clustering for area and power reduction in FPGAs // ACM Trans. Design Autom. Electr. Syst. vol. 7, №4, pp. 643-663. doi:10.1145/605440.605448.

[14] 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.

[15] Gavrilov S. V., Zheleznikov D. A., Chochayev R. ZH. Razrabotka i sravnitel'nyy analiz metodov resheniya zadachi razmeshcheniya dlya rekonfiguriruyemykh 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. – 2020. – T. 25., № 1, S. 48-57., doi: 10.24151/1561-5405-2020-25-1-48-57.

[16] Lot 7. OKR «Razrabotka i osvoyeniye proizvodstva radiatsionno-stoykoy otkazoustoychivoy PLIS yemkost'yu ne meneye 250 tys. logicheskikh ventiley so vstroyennymi blokami PLL i umnozhitelyami», shifr «Almaz-14». (Lot 7. Design and development work "Development and development of production of a radiation-resistant fault-tolerant FPGA with a capacity of at least 250 thousand logic gates with built-in PLL blocks and multipliers", code "Almaz-14"). URL: https://zakupki.gov.ru/epz/order/notice/ok44/view/documents.html?regNumber=0173100009515000200 (дата обращения: 26.08.2022)

[17] ProASIC3 FPGA Fabric User's Guide URL: https://www.microsemi.com/document-portal/doc_download/130708-proasic-sup-u-plus-u-sup-flash-family-fpgas-datasheet (дата обращения: 26.08.2022)

[18] Karpov, S. Flash-semeystva PLIS "Aktel" (Flash-families of FPGA "Actel") / S. Karpov // Komponenty i tekhnologii. – 2007. – № 11(76). – S. 56-62.

[19] Vassiliadis S., Soudris D. Fine- and Coarse-Grain Reconfigurable Computing. Springer, 2007. 381 c.

[20] Karypis G., Kumar V. Parallel multilevel k-way partitioning scheme for irregular graphs // In Proceedings of the 1996 ACM/IEEE conference on Supercomputing (Supercomputing '96), 1996, IEEE Computer Society, USA, p. 21.

[21] Schlag S., et al. k-way Hypergraph Partitioning via n-Level Recursive Bisection URL: https://arxiv.org/abs/1511.03137 (дата обращения: 26.08.2022)

[22] Donath W. Placement and average interconnection lengths of computer logic // IEEE Transactions on Circuits and Systems, vol. 26, no. 4, pp. 272-277, April 1979, doi: 10.1109/TCS.1979.1084635.

[23] Landman B. S., Russo R. L. On a Pin Versus Block Relationship For Partitions of Logic Graphs // IEEE Transactions on Computers, vol. C-20, no. 12, pp. 1469-1479, Dec. 1971, doi: 10.1109/T-C.1971.223159.

[24] Ivannikov, A. D. Formalizatsiya vybora otladochnykh testov pri proyektirovanii tsifrovykh mikroelektronnykh sistem na osnove proverki vypolneniya trebuyemykh funktsiy (Formal Choice of Debugging Tests for Digital Microelectronic Systems Designs Based on the Required Functions Fulfillment Checking) / A. D. Ivannikov, A. L. Stempkovskiy // Izvestiya vysshikh uchebnykh zavedeniy. Elektronika. – 2020. – T. 25. – № 4. – S. 310-319. – doi: 10.24151/1561-5405-2020-25-4-310-319.

[25] D. Shah, E. Hung, C. Wolf, S. Bazanski, D. Gisselquist and M. Milanovic, "Yosys+nextpnr: An Open Source Framework from Verilog to Bitstream for Commercial FPGAs," 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2019, pp. 1-4, doi: 10.1109/FCCM.2019.00010.

[26] Tiunov I.V., Lipatov I.A., Zheleznikov D.A. Development of Methods for Architecturally-oriented Resynthesis in the Computer-aided Design Flow for FPGAs // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2018. Issue 1. P. 69-74. doi:10.31114/2078-7707-2018-1-69-74.