УДК 004.023 DOI 10.31114/2078-7707-2018-1-173-179

Аналитический алгоритм глобального размещения структурированных схем с учетом временных задержек

А.Е. Андреев[1], А.С. Русаков[1,2], А. Яхонтов[1]

[1]eASIC, г. Санта-Клара, rusakov@easic.com

[2]Институт проблем проектирования в микроэлектронике РАН, г. Москва

***Аннотация* — Разработан алгоритм глобального размещения структурированных схем на базе последовательного решения задачи квадратичной оптимизации и легализации на основе потока минимальной стоимости. Применение алгоритма на широком классе реальных задач позволило принципиально улучшить возможности САПР для структурированных схем.**

***Ключевые слова* — выпуклая оптимизация, размещение СБИС, ПЛИС, решение симметричных СЛАУ**

I. Введение

Структурируемые заказные СБИС или схемы, программируемые на фабрике, занимают важную нишу между заказными схемами и ПЛИС и часто экономически наиболее оправданы для реализации функциональности схемы.

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 лишь 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.

Средства САПР, создаваемые для проектирования структурированных схем, должны учитывать как особенности заказных СБИС, так и ограничения, возникающие при проектировании ПЛИС из-за заданной структуры матрицы сайтов. Количество элементов схемы существенно превосходит ПЛИС. Требования к рабочим частотам и энергопотреблению приближаются к возможностям заказных схем. Возможности же САПР существенно ограничены заданной структурой кристалла и ограниченной (в сравнении с заказными схемами) библиотекой элементов, сложностью или даже невозможностью иерархического размещения схемы.

Такие ограничения существенно уменьшают возможности оптимизации схемы на конечных этапах маршрута проектирования. Глобальное размещение структурированной схемы большой размерности фактически определяет возможности дальнейшей оптимизации критических областей и возможности микросхемы.

Увеличение числа элементов, как и в случае заказных схем, требует новых алгоритмов оптимизации размещения. Методы отжига, рекурсивного минимального сечения или же их комбинация [1], которые показывали превосходные результаты, оказываются малоэффективны и не могут найти хорошее глобальное решение. Для заказных схем общепринятым современным подходом являются итерации решения задачи выпуклой оптимизации и последующий быстрой легализации ячеек [2,3]].

Для решения задачи глобального размещения ПЛИС в работе [5] уже предложено использовать подход [2], в работе [4] детально обсуждаются параллельные аспекты. Подход, где формулируется полная нелинейная задача, состоящая из выпуклой части и функций плотности, по нашему опыту для гетерогенных схем труден в настройке из-за необходимости балансировать плотность по каждому ресурсу кристалла.

В статье мы представляем новый алгоритм оптимизации глобального размещения для структурированных схем. Этот алгоритм использует итеративный подход с последовательным решением выпуклой задачи и легализации. Алгоритм применяется на реальных схемах. При разработке нам пришлось решить проблемы, которым в [4,5] фактически не уделяется внимания. Это:

- необходимость улучшения качества оптимизации и времени работы на всех схемах,
- требование автоматического размещения макроблоков памяти,
- высокие требования к максимальным временным задержкам в схемах,
- наличие ячеек, которым в зависимости от плотности соответствуют несколько типов ресурсов.

II. ЗАДАЧА ГЛОБАЛЬНОГО РАЗМЕЩЕНИЯ

Схема представляется в виде гиперграфа *H(V, N)*, где *V*—множество ячеек схемы, *N* - множество гиперребер, описывающих цепи в схеме. Для каждого гиперребра и каждой вершины определен вес $w(n_i)$, $w(v_i)$. 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. Мы также введем для каждой цепи $E(n_i)$ – множество ребер цепи, соединяющие каждые два контакта в цепи. Для цепей с числом контактов больше, чем заранее заданный порог, множество $E(n_i)$ состоит только из ребер от источника цепи. Длина цепи $P(n_i)$ на этапе глобального размещения оценивается как полупериметр прямоугольника, ограничивающего все контакты цепи.

Задача **размещения** ПЛИС или структурированный схемы состоит в назначении логических ячеек схемы на физические сайты так, чтобы выполнялись ограничения на реализацию схемы. Классические ограничения – требования к трассируемости, максимальным временным задержкам, расположению цепей синхронизации.

Задача **глобального размещения** элементов схемы состоит в том, чтобы минимизировать взвешенную сумму длин с весами ребер $w(n_i)$:

$$HPLW = \sum_i w(n_i) * P(n_i),\ i = 1,...,|N|, \qquad (1)$$

расставив ячейки по подобластям так, чтобы в каждой подобласти сумма площадей ячеек не превышала допустимую для данного типа ресурса. Для макроблоков памяти на этапе глобального размещения требуется назначить их на физические сайты.

Полупериметр цепи HPWL – недифференцируемый функционал и труден для оптимизации, поэтому в методах на базе выпуклой оптимизации используют приближения к функционалу HPWL.

Один из вариантов аппроксимации HPWL - это **квадратичный функционал или клика**. Определим

$$\Phi_{clique}(x,y) = \sum_i w(e_i)[(x_k - x_j)^2 + (y_k - y_j)^2], \qquad (2)$$

где $e_i \in E$, E - множество всех ребер $E(n_i)$, (x_j, y_j) и (x_j, y_j) координаты ячеек ребра e_i, $i = 1,...,|E|$. Часть ячеек фиксирована, и их координаты образуют правую часть для (2). Вес ребра, принадлежащего цепи n_k с числом контактов n_{pins}, определяется как:

$$w(e_i) = \frac{w(n_k)}{n_{pins} * (n_{pins} - 1)}. \qquad (3)$$

В работах [2-3] показано, что так называемая B2B (bound to bound) модель дает более точное приближение к HPWL при сравнительно малых движениях ячеек. В **B2B** гиперграф схемы *H(V, N)* разбивается на два графа $G_x(V, E_x)$ и $G_y(V, E_y)$. Для каждого из графов задача решается независимо. Для каждой цепи n_k , состоящей из n_{pins} контактов, находятся граничные контакты с максимальной и минимальной координатами x_r и x_l . Тогда, если ребро цепи e_i связывает ячейку с граничным узлом, определим вес:

$$w_{b2b}(e_i) = \frac{w(n_k)}{(n_{pins} - 1)\min(|x_r - x_l|, \varepsilon)}, \qquad (4)$$

где ε - параметр, ограничивающий минимальную длину ребра. Ребра, соединяющие только внутренние узлы цепи, имеют вес 0. Функционал длины проводов тогда примет вид:

$$\Phi_{b2b}(x,y) = \sum_i w_{b2b}^x(e_i)(x_r - x_l)^2 + \sum_i w_{b2b}^y(e_i)(y_r - y_l)^2. \qquad (5)$$

Итеративный подход, предложенный в [2], состоит в чередовании минимизации решения выпуклого квадратичного функционала и быстрой легализации. На первом этапе итерации решается задача безусловной оптимизации, а все ограничения, включая ограничения на плотность ячеек, учитываются на этапе 2 - этапе легализации. При этом предполагается, что легализация сохраняет относительный порядок ячеек. Для того чтобы внести информацию о легальных позициях в безусловную оптимизацию, вводятся дополнительные псевдо-цепи, соединяющие каждую ячейку и ее вычисленную легальную позицию - якорь. Итак, на каждой итерации *i* на этапе аналитической выпуклой оптимизации решается:

$$\Phi_i(x,y) = \Phi_{awl}(x,y) + \alpha_i \cdot \Phi_{anchor}(x,y), \qquad (6)$$

где α_i - выбранный вес якоря. $\Phi_{awl}(x,y)$ - соответствует либо (2), либо (5), $\Phi_{awl}(x,y)$ - вклад псевдоцепей (якорей). На каждой итерации вес якорей, дополнительных цепей соединяющий ячейку с легальной позицией, увеличивается. При стремлении веса к бесконечности результат решения (6) легален, т.е. удовлетворяет всем ограничениям, и алгоритм оптимизации невыпуклой задачи сходится.

A. Размещение макро-блоков памяти

Принципиальной особенностью и сложностью глобального размещения для технологий eASIC является наличие большого количества макроблоков памяти. Типичное количество физических ячеек памяти 5-10 тысяч, что делает практически

невозможным ручное размещение. Каждая ячейка по площади приблизительно равна 5 тысячам логических ячеек. Высокая утилизация ресурсов памяти в реальных схемах, дискретность допустимых позиций, большое число связей логических ячеек с ячейками памяти делают чрезвычайно сложным инкрементальное переразмещение памяти. Любое движение приводит на этапе легализации к движению большого количества ячеек, скачку длины и функционалов временных задержек. Экспериментальный анализ показал, что одновременное размещение всей схемы с помощью подхода [2] не позволяет достичь хороших результатов за разумное количество итераций.

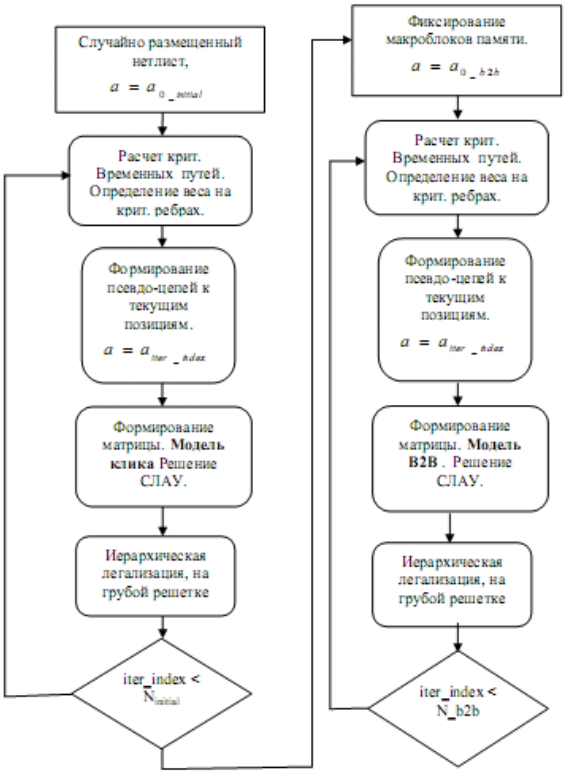

Рис. 1. Схема оптимизации размещения

B. Двухэтапное размещение

Таким образом мы пришли к необходимости двухэтапной реализации алгоритма размещения. На первом ”начальном” этапе в оптимизации учитываются все ячейки, включая ячейки памяти. На этом этапе мы, как и предложено в [2], чередуем выпуклую оптимизацию и легализацию на сравнительно грубой сетке и увеличиваем вес якорей. Фактически, на первом этапе мы автоматически решаем задачу планирования схемы.

После определенного числа итераций ячейки памяти фиксируются, сбрасывается вес якоря и итерационный процесс начинается снова с маленьких весов якорей. В нашем исследовании мы обнаружили, что при первоначальном размещении, когда позиции макроблоков еще не известны, а движения ячеек очень большие, моделирование HPWL цепи с помощью клики ребер (2) приводит к существенно лучшим результатам. На втором этапе необходимо использовать функционал (5). Графическая схема двухэтапного подхода представлена на рис. 1.

C. Решение СЛАУ

Основное время работы алгоритма приходится на формирование и решение СЛАУ на каждой итерации.

$$Ax = b. \qquad (7)$$

Матрица *A*, соответствующая (6), является симметричной положительно-полуопределенной. При положительном весе якорей матрица является положительно-определенной. Благодаря этим свойствам мы можем ожидать, что метод сопряженных градиентов обеспечит сходимость. Трудностью является размерность (порядка 10 миллионов), настройка критерия остановки, плохая обусловленность исходной матрицы на начальных итерациях оптимизации размещения пока якоря вносят небольшой вклад в диагональ матрицы.

Естественным преимуществом метода сопряженных градиентов является возможность получить существенное ускорение решения за счет распараллеливания умножения матрицы на вектор и скалярных произведений.

Чтобы улучшить обусловленность матрицы и настроить итерационный процесс исходная матрица масштабируется так, чтобы получить единичную диагональ:

$$A_s = SAS^t,\ b_s = S\,b\,S^t, \qquad (8)$$

где

$$S = \begin{bmatrix} \sqrt{a_{11}} & 0 & 0 \\ 0 & \sqrt{a_{ii}} & 0 \\ 0 & 0 & \dots \end{bmatrix}$$

Мы можем оценить число обусловленности матрицы A_s, используя теорему Гершгорина. Так как A_s соответствует графу, максимально собственное значение порядка $\lambda_{\max} \approx 2$, минимальное собственное значение можно оценить как $\lambda_{\min} >= d$, где . d - описывает диагональное преобладание:

$$d = \min_i (a_{ii}^s - \sum_{k!=i} |\, a_{ik}^s \,|).$$

Тогда мы можем получить критерий останова. При требовании, чтобы ошибка удовлетворяла условию:

$$\max_i (|\, x_i - \hat{x}_i \,|) < \varepsilon,$$

где $\hat{x}$ - решение, ε - желаемая точность, можно остановить итерационный процесс сопряженных градиентов, если для значений невязки *r* выполняется:

$$\max_i (|\, r_i * a_{ii} \,|) < \varepsilon * \sigma,$$

где $\sigma = \frac{\lambda_{\max}}{\lambda_{\min}}$ оценка числа обусловленности.

Следующим естественным шагом для ускорения алгоритма является предобуславлиние СЛАУ (см. [6]). Надо отметить, что масштабирование (8) равносильно применению диагонального предобуславливателя. Для того, чтобы получить дальнейшее ускорение процесса оптимизации, необходимо выбрать метод предобуславливания, который не должен ухудшать параллельные свойства алгоритма, допускать быстрое построение, существенно уменьшать количество необходимых итераций.

Так как матрица соответствует графу СБИС, естественно ожидать, что в матрице присутствуют слабосвязанные блоки. Для построения предобуславливателя мы можем выделить эти блоки и проигнорировать все внеблочные элементы. Построение и применение матрицы такого предобуславливателя можно делать независимо для каждого подблока и обеспечить необходимые параллельные свойства. Подобный подход для IC2 с перекрытием подблоков рассмотрен в [7], где он назван BJIIC. Мы отфильтровываем из предобуславливателя малые внедиагональные элементы, для которых $|a_{ij^*}\ a_{ij^*}| < \tau \mid a_{jj} * a_{ii} \mid$, соответствующие цепям с большим количеством контактов, улучшив скорость работы и сходимость итерационного процесса.

Мы разработали блочные параллельные варианты метода симметричной верхней релаксации (SSOR) и неполного разложения Холецкого (IC0) для масштабированной матрицы. SSOR не дает заметного выигрыша числа итераций. Блочный IC0, следуя терминологии [7], назовем его BJIIC0, сокращает число итераций приблизительно в 5 раз, при этом стоимость каждой итерации CG увеличивается примерно вдвое.

Эксперименты показали, что на первом этапе оптимизации размещения, когда цепь моделируется кликой, удается использовать разложение IC0 в течение нескольких итераций оптимизации размещения и получать существенный выигрыш, несмотря на то, что исходная матрица A меняется из-за изменения весов якорей и изменения функционала задержек. На втором этапе при моделировании цепи B2B моделью необходимо построение IC0 на каждой итерации оптимизации (6), и выигрыш теряется.

Использование прямых методов для решения полученной СЛАУ также возможно, но эксперименты показали, что скорость работы их примерно в два раза меньше, чем даже последовательные варианты метода сопряженных градиентов.

D. Быстрая иерархическая легализация

В работах [2-4] для этапа быстрой легализации (rough legalization) предлагается использовать варианты геометрической сортировки. Эта сортировка сохраняет относительный порядок ячеек и обеспечивает легальную плотность ячеек в подобластях выбранного масштаба. Наши эксперименты показали, что такой подход не дает удовлетворительных результатов для размещения на матрице структурированного кристалла. Результаты оптимизации оказывались заметно хуже, чем в уже разработанных алгоритмах глобального размещения. Мы объясняем это как принципиальной дискретностью допустимых позиций для ячеек каждого типа, так и наличием нескольких тысяч макроблоков памяти и дискретностью их позиций.

В качестве метода легализации мы выбрали поиск потока минимальной стоимости. Он позволяет найти легальное назначение ячеек на подобласти с минимальным движением. Использование естественной матричной иерархии допустимых позиций на кристалле сильно снижает размерность и одновременно допускает параллельное назначение. Также назначение ячеек каждого типа можно производить независимо и параллельно.

Геометрическая сортировка выполняется уже в маленьких подобластях порядка 100 на 100 микрон и дает хорошее приближение к размещению, получаемому в детальном размещении и упаковке логических элементов на физические сайты кристалла.

E. Оптимизация задержек цепи

При замене алгоритмов на базе метода отжига нам пришлось сравнивать новый подход не только по значению длины цепей, но и по значению худшей и суммарной задержки цепи. Выяснилось, то если недооптимизировать задержки цепи, то общее время в маршруте проектирования может катастрофически вырасти. Это объясняется тем, что оптимизация задержек сдвигается на другие более точные, но и затратные этапы. Метод отжига очень хорошо подходит для локальной оптимизации задержек. На каждом шаге метода отжига размещение легально и вычисленные задержки близки к реальным. Движения локальны, и пересчитывать временные задержки оптимизируемой цепи можно сколько угодно часто. При решении же задачи выпуклой оптимизации (6) находится глобальное решение, и двигаются все ячейки. Длины временных путей, задержки и критичность цепей при этом меняются скачкообразно.

Для оптимизации задержки мы использовали подход модификации весов ребер цепей в зависимости от вклада ребра в функционал задержки цепи. Для каждого входного контакта ячейки на временных путях мы строим ребро или, что тоже самое, двухпиновую цепь, от драйвера цепи до входного контакта. Это ребро входит в уравнение с весом $w_{ti}(sl_{ti}) \in [0, W_{\max}]$. Оптимизируемый функционал (6) примет вид:

$$\Phi_i(x,y) = \Phi_{awl}(x,y) + \alpha_i \cdot \Phi_{anchor}(x,y) + \Phi_{time}(x,y), \quad (6')$$

где

$$\Phi_{time}(x,y) = \sum_{ti} w_{ti} * w_{awl}(e_{ti})[(x_k - x_j)^2 + (y_k - y_j)^2],$$

e_{ti} - ребро соединяющие выходной контакт ячейки с входным контактом на временном пути.

Определим критичность ребра по параметру *c* как относительное расположение ребра в списке, отсортированному по этому параметру, $c \in [0,1]$. Наиболее критичные ребра имеют значения 1. Вес w_{ti} зависит от задержки ребра и критичности ребра с точки зрения влияния на суммарную задержку $c\_wns(e_{ti})$ и худшую $c\_tns(e_{ti})$. Перед итерацией выпуклой оптимизации с помощью методов статического временного анализа (STA) для каждого ребра вычисляется задержка (slack) s_{ti} и количество конечных точек k_{ti} отрицательных временных путей (ending points), чувствительных к данному ребру.

$$w_{ti} = \beta * F(c\_wns_{ti}) + (1-\beta)F(c\_tns_{ti}),$$

$F(c)$ - гладкая монотонная функция, отображающая отрезок $[0,1]$ на $[0,W_{\max}]$. Для более монотонного поведения функционалов суммарной и худшей задержки в процессе итераций добавляется зависимость от "истории". Вычисленный вес ребра модифицируется, и уже это значение используется в оптимизации:

$$w_{ti}(iter) = \gamma \cdot w_{ti} + (1-\gamma)w_{ti}(iter-1).$$

В наших экспериментах β = 0.5, γ = 0.5.

Отметим, что на этапе глобального размещения до буферизации длинных цепей наиболее адекватной является почти линейная модель задержки (delay) цепи.

F. Критерий сходимости итерационного процесса

В работах [2-4] предложено несколько критериев остановки итераций аналитической оптимизации и легализации. Они основываются на разнице между верхней и нижней границей средней длины. При внедрении алгоритма мы отказались от адаптивного критерия. Оказалось, что даже после сходимости функционала средней длины критические пути определяются наиболее точно и продолжают оптимизироваться. При этом функционалы задержек немонотонно зависят от числа итераций. Так как наш алгоритм является частью полного маршрута проектирования, то с точки зрения общего времени работы эффективнее продолжать итерации до достижения заданного числа N_{inital} и N_{b2b}, даже если эти итерации избыточны для оптимизации функционала длины проводов. Отметим, что при большом весе якоря время работы одной итерации существенно ниже, чем в начале итерационного процесса. Вес якорей α_i вычисляется в зависимости от номера итерации:

$$\alpha_i = \exp[\log(\alpha_0) * (1 - i/N) + \log(\alpha_{end}) * (i/N)],$$

где α_0 , α_{end} - начальные и конечные значения веса якоря, *N* – заранее выбранное число итераций данного этапа. Типичные значения для начального этапа размещения: α_0 ,= 0.001, α_{end} =3.5. Для B2B этапа α_0 ,= 0.1, α_{end} =3.5.

III. Сравнительный анализ

Напрямую сравнить разработанный алгоритм с аналогами [3-5] невозможно, потому что существенно отличаются параметры области размещения и библиотеки элементов. Для экспериментального сравнения мы выбрали разработанную для той же технологии комбинацию алгоритмов рекурсивного сечения и метода отжига. Ниже в таблице 1 приведены результаты сравнения результатов размещения с результатами работы [1]. Кристалл, на котором размещались схемы, соответствует 28нм технологии.

Суммарное число итераций N_{inital} и N_{b2b} порядка 150, что существенно выше, чем в [2-5]. В то же время на каждой итерации при большем числе итераций нет необходимости решать несколько выпуклых задач (5) и, кроме того, метод сопряженных градиентов сходится быстрее. Большее количество итераций позволяет чаще пересчитывать задержки цепи и получать более гладкое поведение общего функционала (6').

Алгоритм позволяет значительно улучшить среднюю длину по сравнению с [1]. Время работы при этом удается улучшить в 5-10 раз.

Таблица 1

Сравнение результатов глобального размещения работы подхода на базе метода последовательных сечений и метода отжига и разработанного алгоритма. Полученные значения: последовательные сечения + метод отжига/разработанного алгоритма

	число ячеек	время оптимизации в секундах	средняя длина (um)	худший слек (ns)	Сумма отрицательных задержек (ns)
test 1	$5*10^6$	19900/2800	112/89	-1.91/-3.14	-2270/-1476
test 2	$3*10^6$	21000/3560	139/88	-1.59/-0.87	-780/-62
test 3	$6*10^6$	23200/3200	101/92	-36/-36	-5351/-2612
test 4	$6.3*10^6$	22400/2900	61/49	-11/-11	-733/-3828

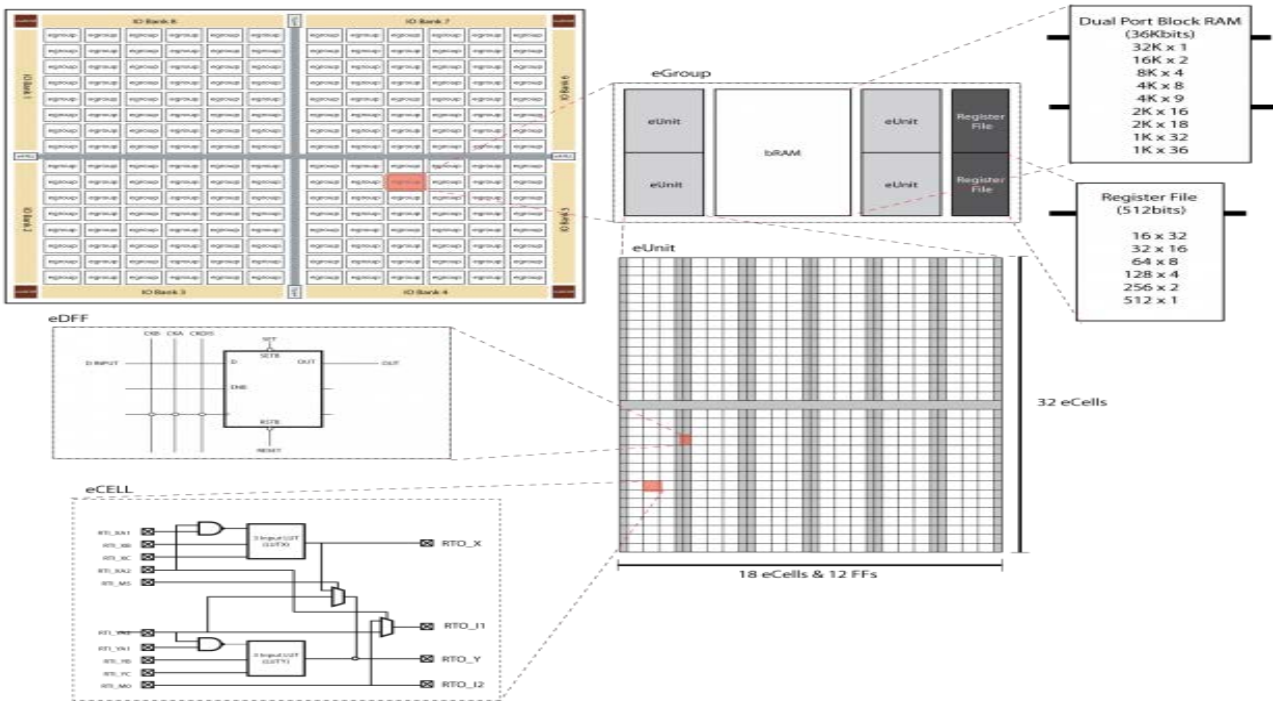

Рис. 2. Принципиальная схема матрицы кристалла Nextreme-2 [8]. Прибор состоит из матрицы групп элементов (eGroups), каждая группа состоит из 4 eUnits, одного элемента памяти Bram и двух регистровых файлов. Каждый eUnit содержит 32*18 ячеек логики eCells и 32*12 регистров

IV. Заключение

В работе представлен параллельный алгоритм глобального размещения схем большой размерности для структурированных интегральных схем.

Алгоритм позволяет значительно сократить среднюю длину и время работы по сравнению с использованием других подходов. Принципиальным отличием разработанного подхода является использование иерархического алгоритма потока минимальной стоимости для легализации двухэтапной схемы оптимизации. Также важным отличием алгоритма является фиксированное и большее число итераций, чем в аналогах, что позволяет не потерять или улучшить качество оптимизации временных задержек в сравнении с локальными методами оптимизации.

Авторы благодарят за чрезвычайно продуктивные дискуссии И. Маркова и А. Никишина.

Литература

[1] Андреев, А. Е., Пависич, И., Русаков, А. С. (2012). Алгоритм глобального размещения структурированных заказных схем. Проблемы разработки перспективных микро-и наноэлектронных систем (МЭС), (1), 231-236.

[2] M.-C. Kim, D.-J. Lee and I.L.Markov, ``SimPL: An Algorithm for Placing VLSI Circuits," Communications of the ACM, vol. 56 no. 6, pp. 105-113, June 2013.

[3] M.-C. Kim and I.L. Markov, ``ComPLx: A Competitive Primal-dual Lagrange Optimization for Global Placement" in Proc. Design Autom. Conf. (DAC), pp. 747-755, San Francisco, CA, 2012.

[4] W. Li, M. Li, J. Wang and D. Z. Pan, "UTPlaceF 3.0: A parallelization framework for modern FPGA global placement," 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Irvine, CA, 2017, pp. 922-928. doi: 10.1109/ICCAD.2017.8203879

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

[6] Y. Saad. 2003. Iterative Methods for Sparse Linear Systems (2nd ed.). Soc. for Industrial and Applied Math., Philadelphia, PA, USA.

[7] И. Е. Капорин, О. Ю. Милюкова, “Неполное обратное треугольное разложение в параллельных алгоритмах предобусловленного метода сопряженных градиентов”, Препринты ИПМ им. М. В. Келдыша, 2017, 037, 28 с.

[8] “eASIC Corporate Website.” http://www.easic.com/. (access date:14.04.2018)

Analytical Timing Driven Global Placement of Structured ASIC

A. Andreev[1], A. Rusakov[1,2], A. Yahontov[1]

[1]eASIC, Santa-Clara, rusakov@easic.com

[2]IPPM RAS,.Moscow

***Abstract* — In the paper we developed an algorithm of structured ASIC global placement based on iteration of solution of convex problem and fast legalization. To achieve high quality of the developed algorithm for structured ASIC technology novel approaches are required in comparison to ASIC state of the art algorithm [3]. Developed algorithm allows obtaining higher quality and faster solution than previously used methodology.**

Physical synthesis CAD tools for structured ASIC are limited in approaches and methods they can use because of technology peculiarities, similar to FPGA CAD limitation. On the other hand, designers demand from structured integrated circuits and physical design tools features close to ASIC. Complexity of the structured ASIC grows. The number of the logical elements, speed of clocks, power consumption requirements significantly exceed FPGA capability.

Global placement of the structured ASIC is the most crucial stage in the physical synthesis flow and for large circuits determines circuit features. New methods for design optimization are required. Min-cut and SA methods can't provide good global solution any more. The state of art approach for ASIC optimization is to use primal-dual optimization approach. An idea originated in SimPL tool [2]. In [2] a solution of the constrained functional is obtained via iterations, on every iteration quadratic convex part and constrained parts are sequentially solved. This approach has already been applied to FPGA placement in [4-5].

In this work we apply ideas from [2-3] for structured ASIC global placement. To achieve production level quality we solved a set of problems which were not yet discussed in previous works. We developed a new fast legalization method based on the network flow, we introduced a two stage global placement to achieve high quality block memory placement, developed a novel parallel linear solution preconditioner, efficient timing optimization scheme is applied.

Developed algorithm shows improvements over previously used method [1] which was very well tuned for the structured ASIC technology. It gives 5-10x speed up, significant wire length decrease and no timing degradation in comparison to [1].

***Keywords* — global placement, VLSI, CAD, structured ASIC, linear system solution.**

REFERENCES

[1] Andreev, Pavisic, Rusakov A.S (2012). Algoritm globalnogo razmechenia structurirovannyh zakaznyck schem. (Algorithm of the global placement of the structed ASICI) / Sb trydov (МЭС), (1), 231-236.

[2] M.-C. Kim, D.-J. Lee and I.L.Markov, ``SimPL: An Algorithm for Placing VLSI Circuits," Communications of the ACM, vol. 56 no. 6, pp. 105-113, June 2013.

[3] M.-C. Kim and I.L. Markov, ``ComPLx: A Competitive Primal-dual Lagrange Optimization for Global Placement" in Proc. Design Autom. Conf. (DAC), pp. 747-755, San Francisco, CA, 2012.

[4] W. Li, M. Li, J. Wang and D. Z. Pan, "UTPlaceF 3.0: A parallelization framework for modern FPGA global placement," 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Irvine, CA, 2017, pp. 922-928. doi: 10.1109/ICCAD.2017.8203879

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

[6] Y. Saad. 2003. Iterative Methods for Sparse Linear Systems (2nd ed.). Soc. for Industrial and Applied Math., Philadelphia, PA, USA.

[7] I. Kaporin, O. Milukova, "Nepolnoe obratnoe treugolnoe razlogenie v parallelnych algorithmach predobyslovlennogo metoda sopryagenych gradientov" (Incomplete inverse triangular factorization in parallel algorithms of preconditioned conjugate gradient methods), IPM Keldysha, 2017, 037, 28 c.

[8] "eASIC Corporate Website." http://www.easic.com/. (access date:14.04.2018) .