файле списки переменных и термов, которые должны находиться сверху или снизу (слева или справа для термов). Эти ограничения обычно не оказывают влияния на эффективность результата. Более сложные ограничения описаны в [4].

3. Можно ограничить работу программы только свертыванием столбцов (или строк) или только свертыванием входов (или выходов).

4. Имеется подпрограмма генерации топологии, позволяющая получить реальную топологию ПЛМ (КМОП или *n*МОП), удовлетворяющую правилам проектирования.

Основные результаты. Описанная программа реализована на языке Си, работает в операционных средах VAX/VMS и UNIX. Основные результаты ее работы для нескольких ПЛМ приведены в таблице.

Средний процент экономии площади матрицы 40. Поскольку топология свернутой ПЛМ имеет ряд особенностей (разрывы сигнальных линий, расположение усилителей входных сигналов сверху и снизу матрицы и т. п.), эта цифра не отражает реальной экономии площади кристалла, однако дает представление об эффективности работы алгоритма.

Программа активно используется при проектировании БИС, наряду с другими средствами оптимизации ПЛМ.

СПИСОК ЛИТЕРАТУРЫ

1. Ульман Дж. Д. Вычислительные аспекты СБИС.—М.: Радио и связь, 1990.
2. Grass W. A depth-first branch and bound algorithm for optimal PLA folding // Proc. 19-th ACM/IEEE Design Automatic Conference.—Las Vegas, Nevada, 1982.—P. 133.
3. Hachtel G. D., Newton A. R., Sangiovanni-Vincentelli A. L. Techniques for PLA folding // Ibid.—P. 147.
4. De Micheli G., Sangiovanni-Vincentelli A. L. PLEASURE: A computer program for simple/multiple constrained/unconstrained folding of PLA // Proc. 20-th ACM/IEEE Design Automation Conference.—Miami Beach, 1983.—P. 530.
5. Липский В. Комбинаторика для программистов.—М.: Мир, 1988.
6. Maciej M. Syslo Discrete Optimization Algorithm with Pascal Program.—Englewood Cliffs, New Jersey: Prentice-Hall, 1983.
7. Емеличев В. А., Мельников О. И., Сарванов В. И., Тышкевич Р. И. Лекции по теории графов.—М.: Наука, 1990.
8. Xuequn Xiang, Saburo Muroga. Interactive reduction of folded PLAs // IEEE.—1986.—P. 592.
9. Sun Young Hwang, Dutton R. W., Blank T. A best-first search algorithm for optimal PLA folding // IEEE Trans. on CAD.—1986.—CAD-5, N 3.
10. Kuo Y. S., Chen C., Hu T. C. A heuristic algorithm for PLA block folding // Proc. 22nd Design Automation Conference.—Las Vegas, Nevada, 1985.—P. 744.
11. Hachtel G. D., Newton A. R., Sangiovanni-Vincentelli A. L. An algorithm for optimal PLA folding // IEEE Trans. Computer-Aided Design.—1982.—CAD-1.—P. 63.

Поступила в редакцию 28 марта 1992 г.

УДК 621.138 : 519.87

Е. Г. Антонянц

(Новосибирск)

АЛГОРИТМ «ПЛУГИРОВАНИЯ» ДЛЯ КОМПАКТИЗАЦИИ ТОПОЛОГИИ СБИС

Представлен алгоритм для модификации топологии СБИС, который можно использовать, с одной стороны, как одномерный компактизатор, с другой — как средство для переорганизации топологии СБИС без изменения электрической схемы, сохраняя при этом корректность по отношению к конструкторско-технологическим ограничениям. Приведены сведения о программной реализации алгоритма «плугирования» и других алгоритмов, существенных для эффективной работы этой операции.

Введение. Процесс проектирования топологии заказных БИС традиционно считается очень трудоемким. Топология современной БИС представляет собой сложный геометрический рисунок, выполненный в нескольких взаимодействующих слоях и составленный из миллионов прямоугольников. Перемещение хотя бы одного из них в процессе модификации топологии может потребовать согласованного перемещения огромного числа других, причем эти переначала топологических проработок. Очевидно, перечисленные выше последствия затрудненной модификации топологии ухудшают в итоге характеристики заказных БИС.

В настоящей статье описывается алгоритм «плугирования» (от слова «плуг»), реализующий мощную интерактивную операцию модификации топологии. Использование «плугирования» дает проектировщику возможность свободно изменять относительное расположение элементов схемы, не задумываясь при этом о сохранении корректности схемы и соблюдении технологических правил проектирования, поддержка которых осуществляется автоматически. Плуг был впервые предложен Джоном Остерхаутом в 1984 г. [1, 2] и реализован как часть системы проектирования интегральных схем "Magic" [3, 4]. При разработке подобной системы в лаборатории автоматизированного проектирования ИС в Институте автоматики и электрометрии СО РАН возникли необходимость и возможность реализации такого средства. В результате был осуществлен алгоритм «плугирования», который представлен ниже.

Суть этой операции ясно иллюстрирует рис. 1. Проектировщик, располагая на топологии плуг — вертикальный или горизонтальный отрезок, задает расстояние и направление движения плуга. Плуг, двигаясь по топологии, выдвигает все объекты, попавшие под плуг, на заданное расстояние таким образом, что все размеры объектов и расстояния между объектами в выбранном направлении уменьшаются до минимально разрешенных технологическими нормами. Таким образом, «плугирование» можно использовать как для компактизации топологии, так и для «расчистки» площади с целью размещения на ней некоторых дополнительных объектов. Следует отметить также, что «плугирование» может использоваться и на уровне геометрии масок (один из немногих примеров одномерных компактизаторов масочного уровня [5]), и на уровне ячеек при работе с иерархически организованными представлениями топологии.

В качестве базы данных в создаваемой системе проектирования интегральных схем ICE [6] применяется структура «тайлового» типа [1]. Подробное ее описание дано в [7]. Необходимо отметить лишь, что интерактивность процедуры «плугирования» обеспечивается за счет высокой скорости работы основных алгоритмов поиска тайлов в такой структуре.

1. **Анализ операции «плугирования».** Предположим, что топология БИС представляет собой множество плоскостей, на каждой из которых сформирован некий геометрический рисунок. Этот рисунок образуется множеством непересекающихся прямоугольников, каждому из которых приписан конкретный тип. «Пустое» место также считается покрытым прямоугольником типа «пусто», и с точки зрения операции «плугирования» эти прямоугольники не отличаются от других. Назовем ребром такой фрагмент вертикальной границы прямоугольника, который разделяет

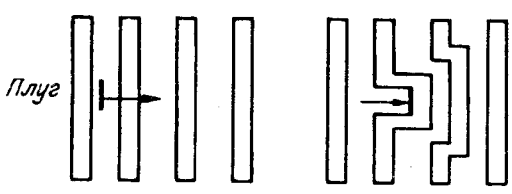


Рис. 1

два различных типа. Элементарным шагом в операции «плугирования» является перемещение одного ребра. Ребро можно представить в виде записи со следующими полями: x — координата по оси x; y, $y1$ — координаты верхнего и нижнего концов ребра; tl — тип материала слева от ребра; tr — тип материала справа от ребра; p — плоскость, которой принадлежит ребро; d — расстояние, на которое передвигается ребро (первоначально для всех ребер величина $d = 0$).

В этом случае для каждого ребра, которое нужно передвинуть (ребра, попавшего в заметаемую плугом площадь), необходимо:

1. Определить область, которая должна быть очищена перед перемещением данного ребра.

2. Проверить, есть ли в этой области какие-либо вертикальные ребра. Если есть, то применить алгоритм «плугирования» рекурсивно для перемещения этих ребер из области «плугирования». Повторять этот шаг, пока все ребра не будут вытеснены из области.

3. Передвинуть начальное ребро на новое место.

Самой сложной и ответственной из этих трех задач является первая. Правильное ее решение гарантирует корректность всей процедуры «плугирования», что включает в себя сохранение электрической связности схемы и соответствие технологическим нормам. Чтобы решить ее, рассмотрим подробнее, как работает плуг. На рис. 2 представлена обычная топология, состоящая из трех несвязанных участков материала. Ребро *e* нужно подвинуть до положения, отмеченного стрелкой. Но прежде чем растягивать материал, который находится слева от ребра *e*, необходимо освободить для него место. Поэтому нужно выдвинуть ребро *f* так, чтобы не возникло новых электрических соединений. Однако нельзя забывать и о технологических нормах. В нашем примере получилось так, что ребро *e* настолько приблизилось к ребру *g*, что перестало выполняться правило минимального расстояния *s* между материалами. Значит, следует отодвинуть и ребро *g*. Другими словами, мы должны очистить и прямоугольник *B*, ширина которого есть минимальное расстояние *s*. Следовательно, перед перемещением любого ребра нужно очистить некоторое пространство, которое в нашем случае есть объединение прямоугольников *A* и *B*. Будем называть его *прямой тенью* ребра.

Кроме того, выдвигая ребро на новое место, нужно следить еще и за тем, чтобы не возникло нарушений правил проектирования в областях сверху и снизу данного ребра. Если выдвигать материал только из прямой тени, как показано на рис. 3 (*2*), то в результате получим разрыв электрической цепи. Поэтому мы должны расширить тень вверх и вниз (в нашем случае на минимально разрешенную ширину материала). Прямоугольники сверху и снизу прямой тени называются *полутенью*. Полутень и прямая тень образуют вместе *тень* ребра. Тень содержит все объекты, которые необходимо выдвинуть из нее перед перемещением данного ребра.

Тогда алгоритм «плугирования» для каждого ребра можно переписать в следующем виде: *1* — найти тень текущего ребра, *2* — очистить эту тень, *3* — передвинуть текущее ребро. Очистка тени включает в себя поиск ребер внутри тени. Для каждого найденного ребра вычисляется новое расстояние и применяется этот же алгоритм «плугирования» ребра.

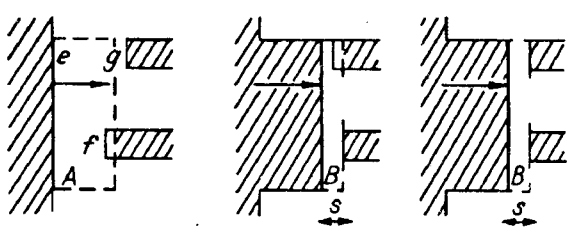


Рис. 2

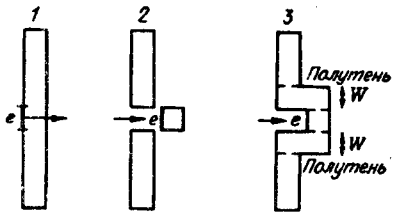


Рис. 3

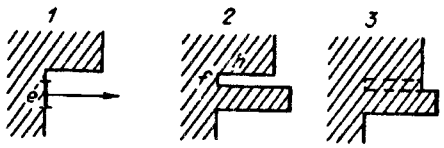


Рис. 4

Таким образом, алгоритм существенно рекурсивен. Конец рекурсии наступает, когда на очередном этапе область, которую нужно очистить перед перемещением текущего ребра, окажется пустой.

Описанная выше процедура «плугирования» исключает случаи нарушения правил проектирования между вертикальными ребрами. Однако она может привести к образованию нарушений технологических норм между горизонтальными ребрами при растягивании материала. Такие нарушения связаны с появлением «лучин» — щелей, образованных материалом или пустым пространством, высота которых меньше разрешенной технологическими нормами. Пример лучины показан на рис. 4.

Для уничтожения лучины необходимо пододвинуть ее дно к меньшей из ее сторон. Так как дно лежит вдоль левой границы полутени, то лучины легко отыскиваются при отслеживании контура полутени. Кроме того, в топологии есть объекты, размеры которых не должны изменяться при модификации проекта. Это относится, например, к транзисторам и контактным окнам. Объявим материалы, образующие такие объекты, нерастяжимыми. Если материал нерастяжим, то левый и правый его края должны двигаться вместе в тандеме. Алгоритм «плугирования» должен учитывать эти ограничения. Описанный выше порядок работы плуга не позволяет ни одному материалу «скользить по другому». Однако в некоторых случаях объекты различных масочных слоев могут перемещаться независимо друг от друга. Металл, например, не взаимодействует ни с поликремнием, ни с диффузией, за исключением контактов.

Чтобы учесть это обстоятельство при реализации операции «плугирования», вся масочная информация о топологии хранится в нескольких невзаимодействующих плоскостях. Материалам одной плоскости разрешается скользить по материалам других плоскостей. Например, *n*МОП-технология может быть представлена с использованием двух плоскостей: одна плоскость содержит металлические проводники, а другая — поликремний, диффузию и транзисторы.

Плуг работает в каждой плоскости независимо. Взаимодействие между плоскостями осуществляется через контакты, которые дублируются в каждой плоскости. Если происходит движение какого-либо края контакта, то соответствующий край этого контакта в другой плоскости тоже должен передвигаться на такое же расстояние.

2. Реализация алгоритма «плугирования». 2.1. *База данных.* Представленный в настоящей статье алгоритм «плугирования» является частью системы обработки топологической информации [6, 7], оперирующей единой базой данных «тайлового» типа [1].

«Тайловые» структуры — это специфический способ представления графической информации манхеттенского типа, позволяющий эффективно реализовать алгоритмы, в которых активно используется понятие «соседства объектов». В такой базе данных вся топологическая информация хранится в нескольких невзаимодействующих плоскостях. Каждая плоскость разбита на неперекрывающиеся прямоугольники — «тайлы», связанные между собой с помощью указателей. На рис. 5 демонстрируется «тайловая» структура и указывается на три важных ее свойства:

1. Каждая точка плоскости принадлежит только одному тайлу, поэтому между объектами одной плоскости не может быть перекрытий. Пустое пространство представляется тайлами особого типа. С этой точки зрения одинако-

выми свойствами обладают тайлы, представляющие какой-нибудь масочный слой, и тайлы, обозначающие пустое пространство.

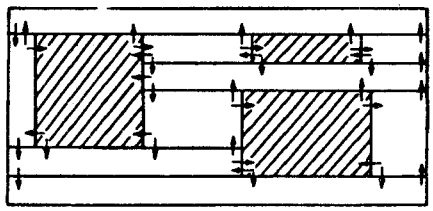


Рис. 5

2. Каждый тайл представлен в виде записей, которые связаны между собой с помощью указателей и образуют таким образом базу данных. Каждый тайл имеет четыре указателя на своих соседей (самого верхнего из правых, самого правого из верхних, самого нижнего из левых и самого левого из нижних).

3. Вся «тайловая» структура имеет горизонтальную или вертикальную ориентацию. Это означает, что в случае горизонтальной ориентации все тайлы максимально вытянуты по горизонтали и, следовательно, не может быть вертикальных границ между тайлами одинакового типа. Это свойство играет решающую роль в алгоритме поиска видимых тайлов, который является характерным для операции «плугирования».

Эта структура хорошо подходит для реализации «плугирования», потому что непосредственно представляет соседствующие объекты топологии и обеспечивает за счет этого линейное время работы следующих операций поиска объектов топологии, необходимых для «плугирования»: 1. Нахождение тайла, содержащего данную точку. 2. Перечисление всех тайлов, лежащих в данной области. 3. Перечисление всех тайлов, граничащих с данным. 4. Перечисление всех тайлов, видимых для данного.

2.2. *Поиск видимых тайлов.* На втором шаге описанного выше алгоритма «плугирования» нужно перечислить все вертикальные отрезки, лежащие в области «плугирования». Здесь удобно использовать алгоритм поиска видимых тайлов. Два тайла взаимно видимы, если между ними можно провести горизонтальную или вертикальную линию, не пересекающую никакие другие непустые тайлы. Горизонтальный поиск видимости основан на алгоритме поиска соседей, который описан в [7]. Следующий алгоритм служит для поиска видимых тайлов с правой стороны начального тайла: 1. С помощью алгоритма поиска соседей перечислить все тайлы, которые касаются правой стороны данного тайла. Для каждого найденного тайла выполняется шаг 2 или шаг 3 в зависимости от типа тайла. 2. Если сосед непустой, то он видим. 3. Сосед — пустой тайл. Если его правая граница продолжается до конца схемы, то такой тайл пропускаем. В противном случае снова с помощью алгоритма поиска соседей перечислим все тайлы, которые касаются правой стороны этого тайла. Каждый из них должен быть непустым. Все тайлы, чьи нижние края лежат ниже верхней границы первоначального тайла, являются видимыми.

Как следует из дальнейшего, временные затраты на операцию «плугирования» существенно зависят от сложности алгоритма нахождения видимых тайлов. Нетрудно построить пример, когда для перечисления видимых тайлов потребуется перебрать все тайлы, входящие в схему. В то же время можно показать [1], что при естественном предположении о равномерности распределения тайлов по площади схемы среднее число видимых из данного отрезка тайлов есть константа, не зависящая от общего числа тайлов в схеме.

2.3. *Двухпроходной алгоритм «плугирования».* Буквальная реализация описанной выше процедуры была бы крайне неэффективной. Каждое ребро, участвующее в «плугировании», передвигалось бы понемногу много раз. Однако есть возможность изменить этот алгоритм так, чтобы его средняя сложность линейно зависела бы от числа участвующих в «плугировании» объектов.

Модифицированный алгоритм состоит из двух последовательных проходов: определения величин сдвигов ребер и собственно передвижения. На первом этапе для каждого ребра вычисляется расстояние, на которое оно будет передвинуто. При этом ребра обрабатываются в «топологическом» порядке. Это означает, что при «плугировании» слева направо полученные ребра

сортируются по координате x и обработка начинается с левого конца списка. Таким образом, каждое ребро не «плугируется» до тех пор, пока не будут обработаны все ребра, которые могут повлиять на его конечное местоположение. На втором этапе ребра передвигаются, изменяя топологию. При этом каждое ребро передвигается один раз.

Первый шаг заключается в том, что вычисляется расстояние, на которое нужно подвинуть первоначальное ребро, и вызываются следующие рекурсивные процедуры для обработки тех ребер, которые должны прийти в движение как следствие перемещения первоначального ребра. Основной алгоритм независим от направления «плугирования» и может быть сформулирован следующим образом: 1. Указатель на первоначальное ребро вставить в голову списка ребер, которые будут передвинуты. 2. Используя расстояние, приписанное данному ребру, и расположение ребра, вычислить площадь, которая должна быть очищена перед перемещением этого ребра, другими словами, найти тень ребра. 3. Используя процедуру поиска видимых тайлов, перечислить все видимые ребра для данного первоначального ребра. Для каждого найденного ребра проделать следующие процедуры: 4. Вычислить расстояние, на которое нужно подвинуть новое ребро, чтобы удалить его из тени первоначального отрезка. 5. Если это расстояние больше, чем то, которое хранится вместе с ребром, то изменить его на большее и поставить новое ребро в очередь для следующей его рекурсивной обработки.

При втором проходе результирующий список ребер сканируется сверху вниз. С каждым ребром списка связано следующее изменение топологии: создается тайл, тип которого совпадает с типом материала слева от текущего ребра, левая его сторона — это текущее ребро, а ширина его — расстояние перемещения, приписанное текущему ребру. Таким образом, материал слева как будто растягивается. Упорядоченность списка ребер гарантирует то, что каждый вновь создаваемый тайл образуется на «чистом месте» и его создание не приведет к нарушению корректности топологии и связности электрической схемы.

Если число передвигаемых тайлов M, а число тайлов в схеме N, то для каждого из двух проходов предполагаемое время работы в среднем пропорционально M. Действительно, при первом проходе рекурсивная процедура вызывается ровно M раз (один раз на каждый передвигаемый тайл). Время работы первого прохода определяется временем, необходимым для перечисления всех видимых соседей для всех передвигаемых ребер. Как отмечалось выше, среднее время поиска видимых для ребра тайлов равно не зависящей от N константе, хотя в худшем случае оно составляет $O(N)$. Для второго прохода время работы определяется временем создания тайла, которое также в среднем является константой [7], поэтому среднее время, затраченное на второй проход, линейно по M. Однако и здесь в худшем случае время создания тайла может быть пропорционально размеру схемы, и тогда сложность второго прохода становится порядка MN.

Заключение. Программа «плугирования» реализована в операционной среде VAX/VMS. Объем программы на языке программирования Си составляет примерно 2000 строк. Поскольку сложность алгоритма «плугирования», работающего над «тайловыми» структурами, линейно зависит от числа элементов топологии, скорость работы программы оказалась достаточно высокой. Это позволило использовать ее в интерактивном режиме непосредственно в редакторе топологических схем.

СПИСОК ЛИТЕРАТУРЫ

1. **Ousterhout J. K.** Corner-stitching: A data-structuring technique for VLSI layout tools // IEEE Trans. on CAD/ICAS.—1984.—P. 87.
2. **Ousterhout J. K., Scott W. S.** Plowing: Interactive stretching and compaction in Magic // Proc. 21th Design Automation Conference.—New Mexico: IEEE, 1984.
3. **Ousterhout J. K., Hamachi G., Mayo R. N. et al.** Magic: A VLSI layout system // Ibid.
4. **Ousterhout J. K., Hamachi G., Mayo R. N. et al.** The Magic VLSI layout system // IEEE J. Design and Test of Computers.—1985.—2, N 1.—P. 18.

5. Mlynski D. A., Sung C.-H. Layout compaction // Layout Design and Verification. Advances in CAD for VLSI.—1986.—V. 4.
6. Титов Д. Г. Система проектирования топологии интегральных схем ICE.—Новосибирск, 1991.—(Препр. СО АН СССР. ИАиЭ; 464).
7. Лившиц З. А., Титов Д. Г. Алгоритмы работы с тайловыми представлениями топологии СБИС // Автометрия.—1991.—№ 3.

Поступила в редакцию 28 марта 1992 г.

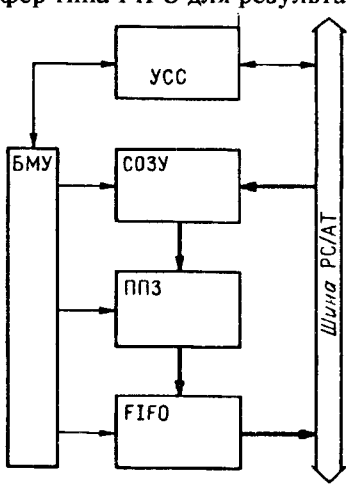
УДК 681.325.538

А. А. Лубков, В. В. Полубинский

(ППЗ) для IBM PC, на порядок увеличивающий производительность систем AT-286, AT-386. Рассмотрена функциональная схема, приведены некоторые временные характеристики.

Введение. Широкое распространение персональных компьютеров типа IBM PC/AT, AT-286, AT-386 и совместимых с ними значительно упростило и ускорило моделирование систем, требующих больших вычислительных ресурсов. В настоящее время у пользователя персонального компьютера больше вычислительных ресурсов, чем у пользователя мини-компьютера в режиме разделения времени. Производительности AT-386 и VAX-780 находятся примерно на одном уровне (1 МФлопс), но VAX-780 должен обслуживать несколько пользователей одновременно. Однако существуют применения, где вычислительная мощность персонального компьютера недостаточна. Таковыми являются алгоритмы цифровой фильтрации и моделирование систем синтеза изображения. Появление быстродействующих БИС отечественного производства для построения процессоров с плавающей запятой позволяет улучшить вычислительные характеристики ПЭВМ и расширить область их применения. БИС серии 1843 подходят для таких целей*.

Функциональная схема. В Институте автоматики и электрометрии СО РАН разработан акселератор операций ППЗ для IBM PC. На одной плате, устанавливаемой непосредственно в PC, размещены (см. рисунок): ППЗ; СОЗУ на 32 слова по 32 разряда; выходной буфер типа FIFO для результатов вычислений; блок микропрограммного управления (БМУ); устройство синхронизации и сопряжения (УУС). Цепочка СОЗУ — ППЗ — FIFO образует конвейер, что позволяет достичь максимальной производительности. Функция, выполняемая акселератором, определяется начальным адресом микропрограммы, загружаемым в акселератор из ЭВМ. Далее программа выполняется линейно, без переходов. Наличие 2 *K* слов микропрограммной памяти позволяет разместить несколько подпрограмм обработки данных (например, подпрограмма перемножения двух матриц [4 × 4] занимает 12 % этой памяти).



Содержимое СОЗУ (загружается ПЭВМ до выполнения операции) при работе не

---

* Сигнальная информация по комплекту микросхем 1843: Завод «Интеграл».—Минск, 1991.