УДК 621.382: 621.3.049.77.001.2 DOI: 10.31114/2078-7707-2020-3-41-48

Ускорение решения задачи обнаружения близости полигонов в задачах технологии двойного шаблона

Д.А. Булах, А.В. Коршунов

Национальный исследовательский университет МИЭТ, г. Москва

dima@pkims.ru, korshun@gmail.com

***Аннотация* — Формирование топологических структур с размерами 22нм и ниже с использованием современного литографического оборудования требует применения ряда техник предварительной подготовки топологической информации, наиболее популярной из которых является технология двойного шаблона. При алгоритмической реализации этой технологии основной вычислительной проблемой является необходимость выполнять большое число сравнений расстояний между различными участками большого числа полигонов. В данной работе предлагается алгоритм сокращения вычислительных затрат, позволяющий обнаруживать только геометрические примитивы, находящиеся в заданной близости.**

***Ключевые слова* — СБИС, топология, двойной шаблон, двойное экспонирование, разреженные матрицы.**

I. Введение

Основным технологическим подходом для физического производства СБИС на данный момент всё ещё является оптическая литография. Несмотря на существование других подходов, таких как, например, электронно-лучевая литография, по ряду причин по-прежнему использование именно оптической литографии является наиболее выгодным [1], [2].

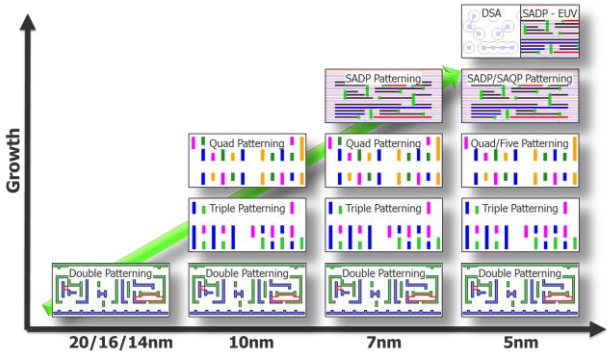

Рис. 1. Кратность применяемых технологий мультиплицирования шаблона в зависимости от технологических норм проектирования

По причинам, связанным с физикой протекания литографического процесса [2], [3], для формирования на поверхности кремниевой пластины изображений требуемого размера при использовании суб-32нм технологий приходится применять ряд дополнительных проектных процедур, наиболее часто используемой из которых является технология мультиплицирования шаблона [4]. В [1] приводится классификация применимости технологий мультиплицирования шаблона в зависимости от техпроцесса, показанная на рис. 1.

Из рис. 1 видно, что наиболее часто применяемой технологией является технология литографии, предполагающая применение нескольких процессов экспонирования и нескольких шаблонов для формирования одного слоя на поверхности кристалла. Различают несколько разновидностей такой технологии [5], [6]:

- двойное экспонирование (англ. Double Exposure Lithography, DEL), также встречающееся под названием LFLE (Litho-Freeze-Litho-Etch), при этом виде технологии выполняется два этапа экспонирования и один этап травления;
- двойной шаблон (англ. Double Patterning Lithography, DPL), также встречающаяся под названием LELE (Litho-Etch-Litho-Etch), при использовании этой технологии травление выполняется два раза.

С точки зрения технологического процесса встречается также подход под названием SADP (Self-Aligned Double Patterning), заключающийся в размещении дополнительного слоя, на котором, в отличие от предыдущих вариантов, формируются не основные изображения геометрических фигур, а формируются фигуры для «прерывания» нанесённых ранее [6].

Поскольку все описанные технологии подразумевают использование двух шаблонов, применяющихся с различными технологическим процедурами, в дальнейшем в этой статье для обозначения подобной технологии будет использоваться термин «двойной шаблон».

Основной задачей технологии двойного шаблона (и в общем случае мультиплицирования шаблона с большей кратностью) является разделение геометрических примитивов топологии, принадлежащих одному слою, на несколько слоёв в случае, если расстояние между этими примитивами меньше некоторого критического размера, определяемого разрешающей способностью литографического оборудования (рис. 2).

Здесь CD – критический размер (англ. Critical Dimension), минимальный размер, который может быть воспроизведён литографическим оборудованием. На рис. 2 слева видно, что имеется 4 конфликта: расстояния межу геометрическими примитивами A и B, A и C, B и

D и C и D меньше критического размера. После разделения топологии на несколько слоёв (рис. 2 справа) все геометрические примитивы в пределах своего слоя расположены на расстоянии, которое может быть воспроизведено.

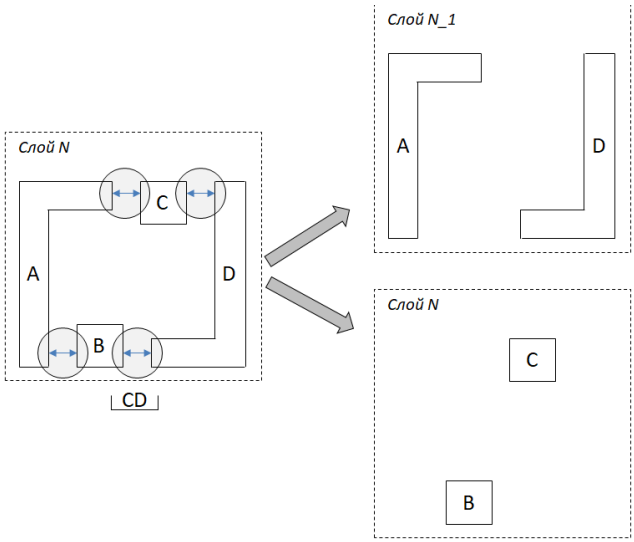

Рис. 2. Разделение топологии на два слоя для компенсации плотности заполнения

В результате выполнения этой операции получаются два слоя, содержащие геометрические примитивы с одного исходного слоя так, что расстояние между примитивами, оказавшимися на одном слое, окажутся больше критического размера и при последовательной смене фотошаблонов изображение сможет быть воспроизведено с требуемой точностью.

Наибольшей проблемой при программной реализации разделения слоя является алгоритмическая сложность решаемой задачи, связанная с большими объёмами данных, требующими обработки.

II. Алгоритмы технологии двойного шаблона

A. Алгоритмы разделения геометрических примитивов на различные слои

Базовым алгоритмом, на основании которого принимается решение о разнесении геометрических фигур на различные слои, является алгоритм раскраски графов [4], [7], [8]. Суть работы алгоритма в следующем. На основании информации о расположенных рядом графических примитивах в пределах одного топологического слоя формируется граф противоречий. Узлы графа ставятся в соответствие примитивам, а рёбра графа показывают необходимость разделения примитивов на разные слои. Другими словами, если два узла графа связаны ребром, это значит, что соответствующие графические примитивы в топологии конфликтуют между собой, так как расположены друг к другу ближе критического размера, и их необходимо разместить в разных слоях. С точки зрения алгоритма, для выполнения этой операции два узла графа, связанные ребром, должны быть раскрашены в разные цвета.

Поскольку при использовании технологии двойного шаблона считается, что для раскраски вершин графа доступны только два цвета, топология может содержать такую конфигурацию, при которой выполнение этой задачи невозможно. Такой пример показан на рис. 3.

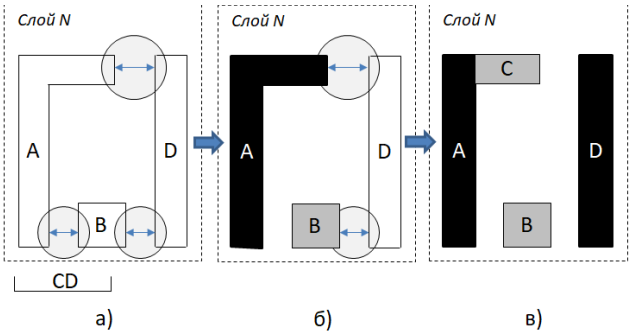

Рис. 3. Иллюстрация необходимости разбиения топологических элементов для успешной раскраски графа противоречий

Для разрешения этого конфликта используется дробление геометрического примитива на части и формирование из одного примитива двух узлов графа противоречий с последующей раскраской в разные цвета. Это приводит к физическому разделению частей одного примитива в разные слои. В этом при формировании фигуры в кремнии выполняется операция межслойной сшивки [6], [9].

На рис. 3-а приведён исходный вариант топологии и показаны три конфликта. Рис. 3-б показывает, что при раскраске примитивов в два цвета (чёрный и серый) одна фигура (D) не может быть окрашена, так как она находится рядом как с фигурой A, покрашенной в чёрный цвет, так и рядом с фигурой B, покрашенной в серый цвет. И только разбиение фигуры A на две (рис. 3-в) позволяет успешно покрасить граф противоречий и разнести топологические элементы на различные слои.

В работах [10], [11] показано, что для 14нм технологии и ниже уже не имеет смысла применять технологию двойного шаблона, а следует применять технологии тройного и четверного шаблона, что алгоритмически вытекает в необходимость раскраски графа противоречий в три и четыре цвета, соответственно.

Однако сам по себе процесс раскраски вершин графа интереса не представляет, поскольку многократно описан и имеет как однопоточные, так и многопоточные реализации, в том числе и с применением технологии распределённых вычислений и вычислений на графических сопроцессорах [12]-[14]. В силу своей большой вычислительной сложности наибольший интерес представляет не алгоритм раскраски графа противоречий, а алгоритм его подготовки, иными словами, алгоритм определения взаимной близости элементов топологии.

B. Алгоритмы формирования графа противоречий

Задача формирования графа противоречий состоит в последовательном обходе всех геометрических фигур топологического слоя с целью выяснения расстояния

между ними. Очевидно, что это – колоссальная по своим вычислительным затратам задача, для реализации которой необходимо вычислить расстояния между всеми участками геометрии всех элементов слоя топологии.

Определить взаимное расположение абсолютно всех геометрических примитивов можно только с помощью полного перебора. Очевидно, что полный перебор – слишком ресурсоёмкая задача, поэтому для сокращения вычислительных затрат применяются различные подходы и алгоритмы.

В работе [15] описан подход, позволяющий существенно сократить объём вычислений при обходе элементов топологии с манхэттенской геометрией, основанный на кодировании линий. Авторами предложен подход, при котором расстояния между различными геометрическими фигурами кодируется целочисленным трёхзначным кодом, однозначно определяющим положение отрезка расстояния в пространстве, а также их взаимное расположение друг относительно друга. Такой подход позволяет хранить информацию о взаимном расположении фигур и на её основе быстро формировать граф противоречий, однако вычисление кодов по прежнему требует обхода всех составляющих всех геометрических примитивов слоя топологии.

В [16] авторы реализуют алгоритм, основывающийся на наложении слоя топологии на сетку с шагом, кратным размерам элементов топологии. Такой подход позволяет легко определить близость различных геометрических фигур, основываясь на обходе ячеек сетки в непосредственной близости от рассматриваемой фигуры. Стоит отметить, что такой метод эффективен, однако требует больших затрат памяти для хранения матрицы и информации о размещении элементов в её ячейках.

Частично эту проблему решают работы [17], [18]. Они посвящены алгоритму для ускорения вычисления расстояний между полигонами а также сокращения затрат памяти на их хранение, который основан на формирования лишь списка горизонтальных отрезков, обрамляющих топологические примитивы слоя. Вычисление расстояний между полигонами осуществляется с применением модифицированного метода сканирующей прямой. Отмечается, что такой подход позволяет эффективно применять средства распараллеливания вычислений. При этом в [17] заявляется возможность приведения неманхэттенской геометрии к манхэттенской с последующим применением описанного выше алгоритма. Тем не менее, в работах отмечается, что для определения горизонтального расстояния между полигонами необходимо проводить дополнительные вычисления на основании точек, составляющих горизонтальные отрезки. Кроме того, в отличие от алгоритма [16] отсутствует возможность для некоторого полигона определить его ближайших соседей иными способами, кроме запуска сканирующей прямой через все полигоны слоя.

В данной работе предлагается структура данных, позволяющая компактно хранить информацию о взаимном размещении полигонов и получить оценку расстояния между любыми произвольными полигонами, а также алгоритмы её заполнения.

III. Предлагаемое решение

Описание разработанного решения в этом параграфе разделено на три подраздела: описание предлагаемого подхода к представлению топологических примитивов, структура данных и алгоритмы для сокращения затрат оперативной памяти, оценка алгоритмической сложности.

A. *Структура данных для компактного хранения информации о взаимном расположении полигонов*

В качестве отправной точки для разработки структуры данных для хранения информации о полигонах была выбрана сеточная модель по типу рассмотренной в работе [15]. С целью ответа на вопрос об эффективности использования памяти рассмотрим тестовый рисунок топологии, представленный на рис. 4.

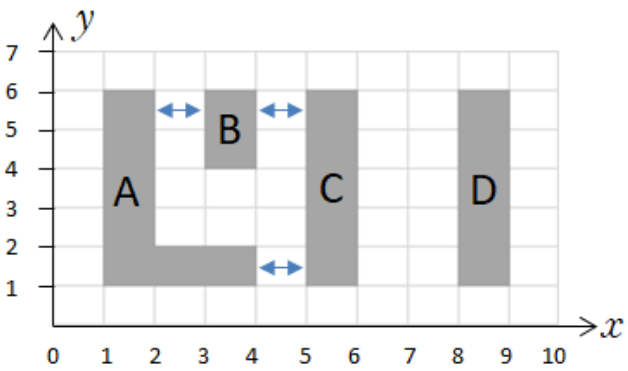

Рис. 4. Тестовый рисунок с тремя обозначенными конфликтами расстояний

Строя классическую сеточную модель для этого примера легко увидеть, что будет выделена память под 70 ячеек, из которых реально занятыми окажутся только 19. Очевидно, что такой подход крайне неэффективен.

Для решения задачи более экономного расходования памяти нами предлагается следующий подход. Каждая фигура F_k определяется следующим кортежем:

$$F_k = \langle M_k, S_k, ref_k \rangle \quad (1)$$

где:

- M_k – условный центр координат k-той фигуры, имеющий элементы $\{m_k^x, m_k^y\}$;
- S_k- метрика оценки её геометрических размеров;
- ref_k – ссылка на топологическое представление самой фигуры.

Для реализации возможны несколько вариантов расчётов условного центра координат M_k и метрики оценки размеров S_k. В настоящей работе центр координат фигуры вычисляется как центр ограничивающего прямоугольника топологической фигуры, а в качестве

метрики оценки геометрических размеров в простейшем случае можно использовать максимальный размер ортогональной проекции на оси системы координат топологии.

Для исходного изображения топологии, приведённого на рис. 4, координаты условных центров координат фигур приведены на рис. 5 (для простоты дальнейших вычислений считается, что все координаты имеют целочисленные значения, для них проводилось округление к ближайшему большему целому).

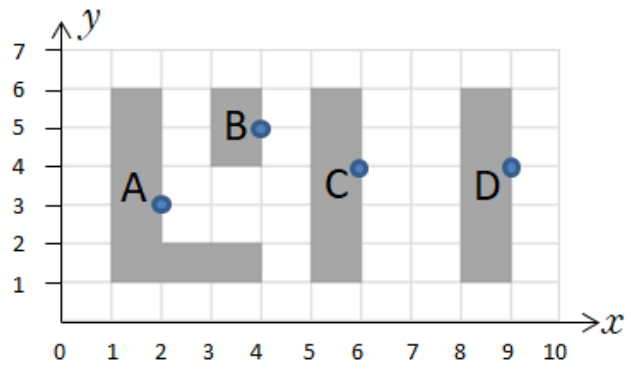

Рис. 5. Координатная сетка с проставленными центрами на целочисленной сетке

Учитывая полученные результаты, для хранения информации об объектах слоя с применением сеточной модели можно получить представление сетки, показанной на рис. 6 (считается, что ячейка занята, если в координаты ячейки попадают координаты условного центра координат фигуры).

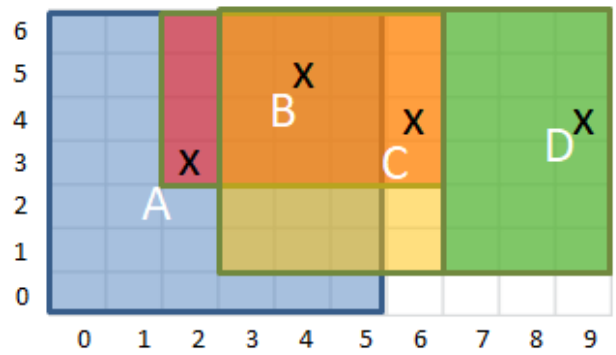

Рис. 6. Сеточная модель топологии с занятыми ячейками (X)

На рис. 6 различными цветами показаны области возможной занимаемой площади, определяемой исходя из выбранной метрики оценки геометрических размеров фигуры. Даже грубая, выбранная ранее метрика (размер максимальный ортогональной проекции), на таком рисунке даёт понять, что фигура D (зелёная область) находится на достаточном расстоянии от фигур A (синяя область) и B (красная область), поэтому при формировании графа противоречий взаимное расположение этих фигур можно не рассматривать.

Введение же в качестве метрики оценки геометрических линейных размеров по каждой из осей по отдельности существенно меняет картину в лучшую сторону (рис. 7).

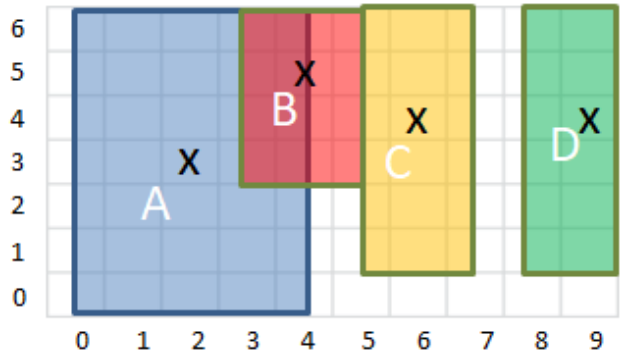

Рис. 7. Вид сеточной модели после переоценки метрики размеров

На данном этапе предложенный вариант реализации сеточной модели пока не даёт выигрыша по занимаемой памяти. Для устранения этого недостатка мы предлагаем обратить внимание на тот факт, что разработанная сеточная модель хранит лишь условные центры координат геометрических фигур, а не занимает фигурой все ячейки пространства топологии, как, например, это делают фигуры на рис. 4. Следовательно, для более компактного хранения такого описания сетки можно воспользоваться технологией хранения разреженных матриц.

B. Структуры данных и алгоритмы для сокращения затрат оперативной памяти

Одним из самых простых форматов хранения разреженных матриц является так называемый Йельский формат. В данной работе применяется Йельский формат хранения «по строкам», также известный как CSR (Compressed Sparse Row) [19].

При использовании этого формата данные о занятых (непустых) ячейках матрицы распределяются между тремя массивами:

- AN – массив данных, его размер равен числу хранимых ненулевых элементов;
- AJ – массив номеров элементов из массива AN, располагающихся на новой строке, его размер равен числу занятых строк разреженной матрицы;
- AI – массив номеров строк для каждого из элементов массива AJ, его размер в точности равен размеру массива AJ.

В терминах разрабатываемого подхода предлагается хранить информацию о фигурах, представленных кортежами вида (1), в следующем виде:

- массив AN хранит координаты x_i условных центров координат каждой из фигур;
- массив AJ хранит информацию, начиная с какого из элементов массива AN произошёл переход на новый ряд;

- массив AI хранит номер ряда (фактически – координата y_i), на котором расположен элемент AN[AJ[i]].

В терминах принятых обозначений матрица, представляющая топологию кристалла, изображённая на рис. 6, будет представлена в формате CSR следующим образом:

AN = {4, 6, 9, 2};

AJ = {0, 1, 3};

AI = {5, 4, 3}.

Добавление элемента в подобную структуру данных представлено в общем виде псевдокодом в листинге 1.

Листинг 1. Код метода Insert_at_XY

```
На входе: M – условный центр координат фигуры
          S – метрика оценки площади
          ref – ссылка на фигуру
 1:  if ∃i: M_y == AI[i]:
 2:      foreach (aj in AJ[i] → AJ[i+1]):
 4:          if M.x < AN[aj].x:
 5:              AN.insert(AN.begin+aj, <M.x, S, ref>
 6:      foreach (aj in AJ[i+1] → AJ.last):
 7:          inc(aj)
 8:  else:
 9:      i = SelectRowByY(M.y)
10 :     AI.insert(AI.begin+i+1)
11:      AJ.insert(AJ.begin+i+1, AJ[i+1])
12:      AN.insert(A.begin+1+1, <M.x, S, ref>)
```

Как видно из листинга, каждому элементу в векторе AN соответствует не только номер столбца матрицы (координата условного центра координат по оси абсцисс), но кортеж вида:

$$AN[i] = \langle M[i].x, S_i, ref_i \rangle \quad (2)$$

Это позволяет, получив доступ к хранимому элементу, получить информацию и о метрике оценки геометрии полигона, и к полному полигону для более точного вычисления координат.

Получение информации о близости полигонов к заданному полигону легко определяется исходя из информации, хранящейся в векторах AN, AJ, AI и (2).

Имея некоторую фигуру топологического слоя F_k, определяемую в оговоренных структурах данных триплетом индексов {k, j, i} (положение фигуры F_kопределяется как AN[k], AJ[j], AI[i]) можно легко определить соседние с ней фигуры. Для этого нужно обратиться к массиву AI с индексами большими и меньшими i до тех пор, пока, значение AI[i] не выйдет за пределы S_k. Для проверки соседей необходимо сдвинуться по индексу k влево и вправо до тех пор, пока AN[k±Δ].x не превышает S_k.

C. Оценка сложности алгоритма

Для оценки сложности алгоритма стоит разобрать наиболее ресурсоёмкие операции, которыми являются:

1. вычисление центров покрывающих полигоны прямоугольников;
2. вычисление высот и ширин прямоугольников, покрывающих полигон;

Для выполнения этих операций определяются минимумы и максимумы среди всех N точек составляющих геометрию фигуры по всем осям.

Имея информацию обо всех углах покрывающего прямоугольника, не представляется сложным нахождение его ширины и высоты, так и получить координаты его центра. Таким образом, сложность вычисления центров покрывающих прямоугольников составляет $O(N)$, сложность вычисления размеров фигур (высоты и ширины) можно не учитывать в силу своей простоты.

Вычисление параметров всех полигонов строится на основе вычисления центров K полигонов, следовательно, сложность вычисления параметров всех полигонов слоя топологии можно оценить как $O(M \cdot K)$, где K – число полигонов, M – среднее число вершин каждого из полигонов.

По сравнению с этими операциями, все остальные операции будут выполняться практически за линейное время.

Таким образом, в среднем время работы предложенных алгоритмов формирования и хранения структуры топологий для последующего формирования графа противоречий можно аппроксимировать как $O(N \cdot K)$, где N – среднее число вершин полигонов на слое, K – число геометрических фигур на слое.

IV. Пример

В качестве примера был обработан топологический слой элемента NAND2. Исходная топология слоя показана на рис. 7-а. На рис. 7-б и 7-в показаны слои после обработки алгоритмом. Рис. 7-б показывает расположение условного центра координат объектов (для наглядности они представлены прямоугольниками достаточно большого размера, чтобы быть заметными на топологии слоя, отмечены синим цветом). На рис. 7-в показан слой с нанесённой областью, показывающей область покрытия фигуры.

В табл.1 приведён пример разреженной матрицы, хранящей информацию об изображении слоя топологии, представленной на рис. 7. Для удобства представления информации вектора расположены вертикально.

На основании представленных в таблице данных можно показать, как для конкретного элемента определить, какие элементы расположены к нему в пределах некоторого значения ε (в случае технологии двойного шаблона – критический размер разрешающей способности). Допустим, требуется установить, какие полигоны расположены в непосредственной близости от полигона P_{11} с точностью в 3.0 мкм. Из таблицы видно, что координаты центра этого полигона в микронах – (19.0, 59.0).

Таблица 1

Сформированная разреженная матрица координат

P#	AN			AJ	AI
	x, мкм	S		index	Y, мкм
		w, мкм	h, мкм		
1	16.5	34.0	11.0	1	11.5
2	16.5	5.0	5.0	2	11.4
3	8.5	4.0	17.5	3	10.825
4	24.5	4.0	18.5	4	10.775
5	8.5	5.0	20.0	5	93.5
6	16.5	5.0	20.0	8	64.0
7	24.5	5.0	20.0	10	61.5
8	11.0	5.0	5.0	11	59.0
9	19.0	5.0	5.0	12	57.25
10	23.25	17.5	52.0	13	36.0
11	19.0	5.0	20.0	14	26.0
12	8.25	4.0	14.0	16	11.5
13	21.5	9.5	7.0	17	6.0
14	8.5	4.0	11.0	18	5.0
15	21.5	5.0	20.0	19	0.0
16	8.5	5.0	5.0		
17	16.5	4.0	17.0		
18	15.0	31.0	11.0		
19	0.0	5.0	5.0		

Для определения ближайших соседей требуется выполнить два шага.

Шаг 1. Определение области интереса по оси ординат. Она составит 59.0 ± (3.0 + ½ · h) мкм, где h – высота рассматриваемого полигона. Таким образом, нас будут интересовать все полигоны, координаты Y которых лежат в диапазоне [46.0 .. 72.0] мкм. Из таблицы видно (последний столбец, хранящий значения по оси ординат, и четвёртый столбец, хранящий высоты полигонов), что центры ограничивающих прямоугольников с поправкой на собственную высоту в этом диапазоне имеют полигоны с индексами от 8 до 12, (P_{11} не учитываем, это рассматриваемый полигон).

Шаг 2. Для выбранных полигонов P_8, P_9, P_{10} и P_{12} требуется проверить координаты по оси абсцисс. Область интереса по оси абсцисс составляет 19.0 ± (3.0 + ½ · w) мкм, где w – ширина покрывающего прямоугольника рассматриваемого полигона, то есть необходимо найти полигоны из указанного списка, координаты которых по оси абсцисс лежат в диапазоне [13.5 .. 24.5] мкм. Из таблицы видно (второй столбец, хранящий значения по оси абсцисс, и третий столбец, хранящий ширины полигонов), что координат с поправкой на собственную ширину в этом диапазоне имеют все перечисленные полигоны, кроме P_{12}, что подтверждается визуально на рис. 8.

Таким образом, при формировании графа противоречий в задаче подготовки двойного шаблона при рассмотрении полигона P_{11} необходимо провести более точные вычисления для проверки близости его граней только с гранями пяти полигонов P_8, P_9 и P_{10}, а не перебирать все грани всех полигонов слоя.

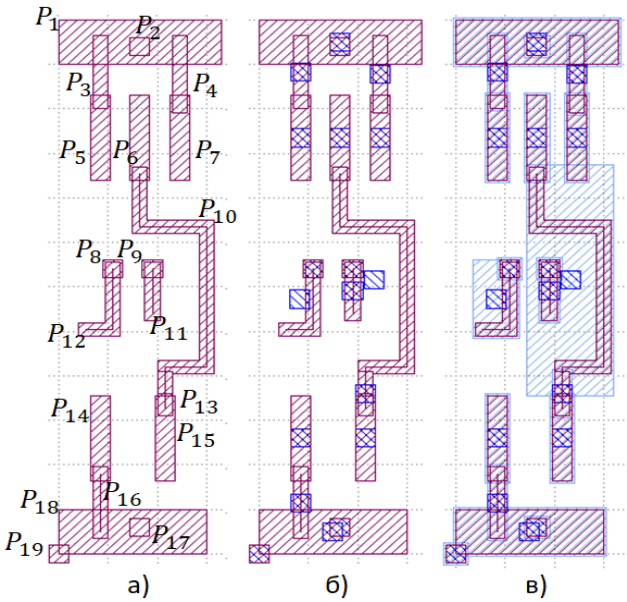

Рис. 8. Тестовый слой топологии элемента NAND2

V. Выводы

В работе представлены разработанный подход, позволяющий в компактном виде представлять информацию о взаимном расположении геометрических элементов для последующего выполнения задачи формирования графа противоречий и применения технологии двойного шаблона, а также описан алгоритм работы с предложенной структурой данных.

Показано, что предлагаемый формат хранения топологии для формирования графа противоречий является более компактным с точки зрения затрат оперативной памяти по сравнению с применяемыми в других рассмотренных алгоритмах.

Считая, что M – среднее число вершин в полигонах слоя, K – число полигонов в слое, большинство рассмотренных в литературных источниках алгоритмов определяют близость в среднем за время $O(M^2 \cdot K)$. Разработанный алгоритм позволяет обойти все полигоны в среднем за время $O(M \cdot K)$, при этом позволяет сэкономить память на хранение информации о расположении полигонов.

Поддержка

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 20-07-00556.

Литература

[1] Adapted from A. Raley et al. “A spacer-on-spacer scheme for self-aligned multiple patterning and integration,” Proc. SPIE 9782, 97820F (2016), URL: https://www.researchgate.net/publication/306088534_A_spacer-on-spacer_scheme_for_self-aligned_multiple_patterning_and_integration (дата обращения 02.04.2020).

[2] David Abercrombie. Will EUV Kill Multi-Patterning? Mentor Graphics Blog, 25.01.2017, URL:

https://blogs.mentor.com/calibre/blog/2017/01/25/will-euv-kill-multi-patterning/ (дата обращения 02.04.2020).
[3] Thiago Rosa Figueiro. Process modeling for proximity effect correction in electron beam lithography. Micro and nanotechnologies/Microelectronics. Université Grenoble Alpes, 2015. English. ffNNT : 2015GREAT011ff. URL: https://tel.archives-ouvertes.fr/tel-01206934/document (дата обращения 02.04.2020).
[4] Y. Ma, X. Zeng and B. Yu, "Methodologies for layout decomposition and mask optimization: A systematic review," 2017 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), Abu Dhabi, 2017, pp. 1-6, doi: 10.1109/VLSI-SoC.2017.8203477.
[5] Lee S, Byers J, Jen K, Zimmerman P, Rice B, et al. 2008. An analysis of double exposure lithography options. Proc. SPIE 6924(Pt. 2):69242A.
[6] D. Z. Pan, J. Yang, K. Yuan, M. Cho and Y. Ban, "Layout optimizations for double patterning lithography," 2009 IEEE 8th International Conference on ASIC, Changsha, Hunan, 2009, pp. 726-729, doi: 10.1109/ASICON.2009.5351308.
[7] Andrew B. Kahng, Chul-Hong Park, Xu Xu and Hailong Yao. Layout Decomposition Approaches for Double Patterning Lithography. IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 29, NO. 6, JUNE 2010.
[8] M. Lin, Y. Li and K. Lin, "Color balancing aware double patterning," 2017 International Conference on Applied System Innovation (ICASI), Sapporo, 2017, pp. 284-287, doi: 10.1109/ICASI.2017.7988407.
[9] Minsik Cho, Yongchan Ban and D. Z. Pan, "Double patterning technology friendly detailed routing," 2008 IEEE/ACM International Conference on Computer-Aided Design, San Jose, CA, 2008, pp. 506-511, doi: 10.1109/ICCAD.2008.4681622.
[10] B. Yu, K. Yuan, D. Ding and D. Z. Pan, "Layout Decomposition for Triple Patterning Lithography," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 34, no. 3, pp. 433-446, March 2015, doi: 10.1109/TCAD.2014.2387840.
[11] A. Lvov, G.E. Téllez, G. Nam. On Coloring and Colorability Analysis of Integrated Circuits with Triple and Quadruple Patterning Techniques. ISPD '18: Proceedings of the 2018 International Symposium on Physical DesignMarch, 2018, pp 152–159.
[12] J. R. Allwright and R. Bordawekar and P. D. Coddington and K. Dincer and C. L. Martin. A Comparison of Parallel Graph Coloring Algorithms. Technical report, Syracuse University, 1995. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.4650 (дата обращения: 15.06.2020).
[13] P. Li et al., "High Performance Parallel Graph Coloring on GPGPUs," 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Chicago, IL, 2016, pp. 845-854, doi: 10.1109/IPDPSW.2016.11.
[14] A. E. Sariyüce, E. Saule and Ü. V. Çatalyürek, "Scalable Hybrid Implementation of Graph Coloring Using MPI and OpenMP," 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, Shanghai, 2012, pp. 1744-1753, doi: 10.1109/IPDPSW.2012.216.
[15] Khaled M. Soradi Nevin M. Darwish. Enhanced double patterning decomposition using lines encoding. Journal of Electrical Systems and Information Technology, Vol. 3, Issue 2, September 2016, pp. 210-216.
[16] K. Yuan, J.-S. Yang, and D. Z. Pan, Double patterning layout decom-position for simultaneous conflict and stitch minimization, inProc. Int.Symp. on Physical Design, March 2009.
[17] Верстов В.А. Высокопроизводительный алгоритм восстановления графовых моделей представления топологии сбис для технологии двойного шаблона. Молодёжный научно-технический вестник – Москва.: Издательство МГТУ им. Н.Э. Баумана, - Эл. журнал, № 10, сентябрь 2012. URL: http://ainsnt.ru/doc/482845.html (дата обращения: 115.06.2020).
[18] Верстов В.А. Параллельный алгоритм трансформации топологического слоя цифровой схемы при наличии ограничений технологии двойного шаблона: Автореф… дис. кан. техн. наук. – М.: 2016. – 16 с.
[19] F. Smailbegovic, G. Gaydadjiev, S. Vassiliadis, Sparse matrix storage format, in: Proceedings of the 16th Annual Workshop on Circuits, Systems and Signal Processing, 2005, pp. 445–448.

Speeding up Evaluation of Polygons Mutual Placement Task for Double Pattern Technology

D.A. Bulakh, A.V. Korshunov

National Research University of Electronic Technology, Moscow

dima@pkims.ru, korshun@gmail.com

***Abstract* — Manufacturing ICs with scale less than 22 nm using optical lithography requires the use of a number of special techniques for preliminary preprocessing layout information and the most used in nowadays is double patterning technology (DPT). Implementation of this technique requires to solve the main problem: how to compare all the distances between all the polygons within given layout. In this paper we propose an algorithm which is intended to reduce computations and allowing to detect only polygons that are in a required proximity.** **Funding: The reported study was funded by RFBR, project number 20-07-00556**

***Keywords* — VLSI, layout, double patterning, double exposure, sparse matrices**

References

[1] Adapted from A. Raley et al. “A spacer-on-spacer scheme for self-aligned multiple patterning and integration,” Proc. SPIE

9782, 97820F (2016), URL: https://www.researchgate.net/publication/306088534_A_spacer-on-spacer_scheme_for_self-aligned_multiple_patterning_and_integration (access date 02.04.2020).
[2] David Abercrombie. Will EUV Kill Multi-Patterning? Mentor Graphics Blog, 25.01.2017, URL: https://blogs.mentor.com/calibre/blog/2017/01/25/will-euv-kill-multi-patterning/ (access date 02.04.2020).
[3] Thiago Rosa Figueiro. Process modeling for proximity effect correction in electron beam lithography. Micro and nanotechnologies/Microelectronics. Université Grenoble Alpes, 2015. English. ffNNT : 2015GREAT011ff. URL: https://tel.archives-ouvertes.fr/tel-01206934/document (access date 02.04.2020).
[4] Y. Ma, X. Zeng and B. Yu, "Methodologies for layout decomposition and mask optimization: A systematic review," 2017 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), Abu Dhabi, 2017, pp. 1-6, doi: 10.1109/VLSI-SoC.2017.8203477.
[5] Lee S, Byers J, Jen K, Zimmerman P, Rice B, et al. 2008. An analysis of double exposure lithography options. Proc. SPIE 6924(Pt. 2):69242A.
[6] D. Z. Pan, J. Yang, K. Yuan, M. Cho and Y. Ban, "Layout optimizations for double patterning lithography," 2009 IEEE 8th International Conference on ASIC, Changsha, Hunan, 2009, pp. 726-729, doi: 10.1109/ASICON.2009.5351308.
[7] Andrew B. Kahng, Chul-Hong Park, Xu Xu and Hailong Yao. Layout Decomposition Approaches for Double Patterning Lithography. IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 29, NO. 6, JUNE 2010.
[8] M. Lin, Y. Li and K. Lin, "Color balancing aware double patterning," 2017 International Conference on Applied System Innovation (ICASI), Sapporo, 2017, pp. 284-287, doi: 10.1109/ICASI.2017.7988407.
[9] Minsik Cho, Yongchan Ban and D. Z. Pan, "Double patterning technology friendly detailed routing," 2008 IEEE/ACM International Conference on Computer-Aided Design, San Jose, CA, 2008, pp. 506-511, doi: 10.1109/ICCAD.2008.4681622.
[10] B. Yu, K. Yuan, D. Ding and D. Z. Pan, "Layout Decomposition for Triple Patterning Lithography," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 34, no. 3, pp. 433-446, March 2015, doi: 10.1109/TCAD.2014.2387840.
[11] A. Lvov, G.E. Téllez, G. Nam. On Coloring and Colorability Analysis of Integrated Circuits with Triple and Quadruple Patterning Techniques. ISPD '18: Proceedings of the 2018 International Symposium on Physical DesignMarch, 2018, pp 152–159.
[12] J. R. Allwright and R. Bordawekar and P. D. Coddington and K. Dincer and C. L. Martin. A Comparison of Parallel Graph Coloring Algorithms. Technical report, Syracuse University, 1995. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.4650 (access date: 15.06.2020).
[13] P. Li et al., "High Performance Parallel Graph Coloring on GPGPUs," 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Chicago, IL, 2016, pp. 845-854, doi: 10.1109/IPDPSW.2016.11.
[14] A. E. Sariyüce, E. Saule and Ü. V. Çatalyürek, "Scalable Hybrid Implementation of Graph Coloring Using MPI and OpenMP," 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, Shanghai, 2012, pp. 1744-1753, doi: 10.1109/IPDPSW.2012.216.
[15] Khaled M. Soradi Nevin M. Darwish. Enhanced double patterning decomposition using lines encoding. Journal of Electrical Systems and Information Technology, Vol. 3, Issue 2, September 2016, pp. 210-216.
[16] K. Yuan, J.-S. Yang, and D. Z. Pan, Double patterning layout decom-position for simultaneous conflict and stitch minimization, inProc. Int.Symp. on Physical Design, March 2009.
[17] Verstov V.A. Visokoproizvodutel'niy algoritm vosstanovlenija grafovykh modeley predstavleniya topologii SBIS dla tekhnologii dvoynogo shablona (High-performance algorithm for graph models reconstructing of the VLSIs layout representation for double patterning technology) // Molodezhniy nauchno-tekhnicheskiy vestnik – Moskva.: Izdatel'stvo MGTU im. Baumana, - El. zhurnal, № 10, sentjabr' 2012. URL: http://ainsnt.ru/doc/482845.html (access date: 15.06.2020).
[18] Verstov V.A. Parallel'niy algoritm transformacii topologicheskogo sloya cifrovoy shemi pri nalichii ogranicheniy tekhnologii dvoynogo shablona (Parallel algorithm for digital circuit layout transformation with the presence of double patterning technoology limitations) // Avtoref. dis. kan. tehn. nauk. – M.: 2016. – 16 p.
[19] F. Smailbegovic, G. Gaydadjiev, S. Vassiliadis, Sparse matrix storage format, in: Proceedings of the 16th Annual Workshop on Circuits, Systems and Signal Processing, 2005, pp. 445–448.