УДК 519.688

Автоматический вывод дополнительных ограничений на границах стандартных ячеек

С.А. Быков, Н.В. Рыженко, А.А. Сорокин

АО «Интел А/О», sergey.bykov@intel.com

Аннотация **— В настоящей работе рассматривается решение для исследования и разработки дополнительных ограничений на границах стандартных ячеек. Данная работа актуальна в решении проблем вычислительной геометрии при проектировании топологий стандартных ячеек. В рамках данной задачи необходимо спроектировать электронный компонент СБИС – стандартную ячейку – таким образом, чтобы ее топология соответствовала заданным геометрическим ограничениям и ограничениям на электрические параметры. В процессе проектирования вводятся дополнительные правила, которые позволяют избегать нарушений, возникающих при составлении ряда ячеек. В работе используются методы решения задачи выполнимости булевых функций, перечисления полных подграфов и минимизации логических функций.**

Ключевые слова **— трассировка, стандартные ячейки, правила проектирования, библиотека элементов, оптическая литография, булева выполнимость.**

I. Введение

Подход с использованием стандартных ячеек способствует уменьшению сложности и сокращению ручного труда при проектировании интегральных схем (ИС). Стандартные ячейки позволяют абстрагироваться от деталей реализации той или иной логической функции, что дает возможность одним разработчикам сфокусироваться на высокоуровневых аспектах дизайна, в то время как другие работают над физическими реализациями ячеек. Библиотеки стандартных ячеек составляются таким образом, что правила проектирования не нарушаются при любом разрешенном размещении ячеек.

Проблема трассировки принадлежит к классу NP-полных задач [1]. Существуют различные подходы трассировки на уровне функциональных блоков. В частности, в работе [2] рассматривается канальная трассировка. Волновой алгоритм Ли представлен в работах [3, 4]. Инкрементальный подход широко используется при решении задачи трассировки и включает в себя многократную перетрассировку одной и той же цепи. Такое решение способствует достижению необходимого качества трассировки и вместе с тем завершает все цепи.

Традиционные алгоритмы трассировки не обеспечивают необходимого результата в случае стандартных ячеек. Как правило, стандартная ячейка представляет собой крайне перегруженную область, что ограничивает применимость алгоритмов трассировки блок-уровня. Большое число сложных геометрических правил проектирования (англ. design rules) также обуславливает неудовлетворительные результаты итеративных методов трассировки. В качестве решения данной проблемы рассматриваются алгоритмы параллельной трассировки [5-11].

В силу большого числа и сложности правил проектирования затруднительно сформулировать дополнительные ограничения на границах, предотвращающие нарушения технологических норм, которые могут возникнуть между соседними ячейками. Сложно формализовать и процесс построения таких правил — каждый технологический процесс или новый набор правил требует нового подхода или алгоритмов.

Множество стандартных ячеек, сгенерированных с учетом заданных ограничений на границах, должно соответствовать определенным требованиям по площади, энергопотреблению и задержкам. Необходимость найти субоптимальный набор правил представляет значительную сложность — предугадать, как именно правила повлияют на эти свойства ячеек, затруднительно. Более того, трудно сформулировать целевую функцию, которая позволила бы выбирать между наборами правил.

Данная работа предлагает программный комплекс автоматического вывода вспомогательных ограничений на границах, которые затем используются при синтезе стандартных ячеек.

Целью работы является автоматизация процесса построения вспомогательных ограничений при заданных правилах проектирования. Дополнительные правила включают в себя границы ячеек и позволяют избегать любых нарушений, возникающих между соседними ячейками. Стандартные ячейки, созданные с учетом полученных ограничений, удовлетворяют определенным требованиям к плотности, задержкам, энергопотреблению и др.

II. Получение дополнительных ограничений

A. Постановка задачи

Входными данными является множество правил проектирования, которое запрещает определенные комбинации объектов топологии.

Каждый объект топологии может быть размещен в определенных 2D точках трассировочной сетки; правила проектирования при этом формулируются в виде булевых функций [12].

Модель данных, описанная в работе [12], опирается на использование трассировочной сетки. В каждом пересечении горизонтальной и вертикальной линии сетки могут быть размещены различные объекты топологии. Сетка задает фиксированные дискретные позиции и типы объектов, которые могут быть в них размещены. Каждому объекту соответствует булева переменная: если объект присутствует, то ее значение «1» («истина»), если отсутствует — «0» («ложь»).

Исходная проблема синтеза формулируется следующим образом: необходимо найти такое множество объектов топологии в ограниченной области, которое бы удовлетворяло всем заданным ограничениям и правилам проектирования. Синтезированные стандартные ячейки далее используются при построении функциональных блоков. Стоит отметить, что на этапе синтеза невозможно предугадать, какие ячейки окажутся рядом с текущей. Без этого контекста две стандартных ячейки без нарушений правил проектирования, поставленные рядом, все еще могут образовать запрещенную топологию.

На рис. 1. приведен пример подобного нарушения. Объекты x_i и x_j не могут быть поставлены рядом друг с другом, в соседних узлах (рис. 1, а, б). В ячейке слева есть объект x_2, в ячейке справа пара объектов x_6 и x_8. Граница ячеек отмечена жирной линией, штрихованные прямоугольники выделяют 2 нарушения заданного правила.

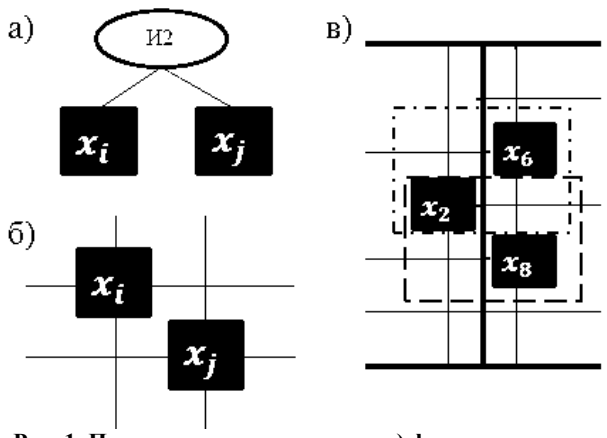

Рис. 1. Правила проектирования: а) формула правила [12], б) запрещенный набор объектов, в) два возможных нарушения на границе

Несмотря на то, что обе ячейки не содержат нарушений сами по себе, вместе они производят конфликтную ситуацию. Пары объектов x_2x_6 и x_2x_8 запрещены (рис. 1, в).

В. Поиск всех решений

1) Описание правил проектирования. В данной работе используются наработки в области решения задачи разрешимости булевых функций (англ. SAT) [13] для нахождения разрешенной топологии при заданных правилах проектирования. SAT позволяет формулировать произвольные отношения между проводниками и межслойными переходами в форме булевых выражений.

В рамках задачи выполнимости булевых функций проводится поиск такой комбинации значений переменных, входящих в булеву формулу $F(x_0, \dots, x_n)$, которая ей удовлетворяет. Например, рассмотрим формулу:

$$F(a, b) = a \wedge \neg b. \quad (1)$$

Значения «1» и «0» могут быть заданы булевым переменным a и b. Значения $a|1$, $b|0$ удовлетворяют формулу (1), с другой стороны, значение $a|0$ не удовлетворяет (1).

Большинство известных программ поиска решений принимают на вход описание булевой функции в конъюнктивной нормальной форме (КНФ). Формула, построенная с использованием законов де Моргана, двойного отрицания и дистрибутивного закона, может потребовать 2^n дополнительных дизъюнкций. В литературе рассматриваются различные способы представления выражений логики высказываний в форме КНФ [14, 15]. В данной работе используется преобразование Цейтина [14], которое вводит дополнительные переменные в исходное логическое выражение.

Рассмотрим пример, изображенный на рис. 1, а, б, который запрещает одновременное присутствие 2 объектов в диагональном направлении:

$$R\left(x_i, x_j\right) = \mathrm{x_i} \wedge x_j. \quad (2)$$

В случае, когда оба объекта x_i и x_j присутствуют, формула (2) возвращает значение «истина». Данное ограничение может быть представлено логическим вентилем И. Любой вентиль может быть представлен в виде КНФ с использованием дополнительных переменных. В данной работе вспомогательная переменная out соответствует выходу логического вентиля. Несмотря на то, что полученная формула содержит больше переменных, она может быть удовлетворена теми же значениями входных параметров, что и исходная. Таким образом, полученная КНФ формула возвращает значение «истина» тогда и только тогда, когда удовлетворено и исходное выражение логики высказываний.

Представим (2) в виде КНФ:

$$P\left(out, x_i, x_j\right) = \left(out \vee \neg x_i \vee \neg x_j\right) \wedge (\neg out \vee x_i) \wedge$$
$$\wedge (\neg out \vee x_j). \quad (3)$$

Каждый набор значений переменных x_i и x_j соответствует строчке со значением «истина» в таблице истинности оператора И. Например, при значениях

$out|1, x_i|1,\ x_j|1$ или $out|0,\ x_i|0, x_j|1$ формула P возвращает значение «1».

Формула (3) может быть удовлетворена любыми значениями x_i и x_j. Но исходное ограничение запрещает определенные комбинации значений для этих переменных. Отрицание формулы (3) будет описывать только запрещенные наборы значений. Инверсия переменной out соответствует выражению $\neg P\ (out, x_i, x_j)$:

$$\neg P\ (out, x_i, x_j) = \neg out \wedge (out \vee \neg x_i \vee \neg x_j) \wedge$$

$$\wedge (\neg out \vee x_i) \wedge (\neg out \vee x_j). \quad (4)$$

Если формула (4) удовлетворена, то значения x_i и x_j могут интерпретироваться как топология без нарушения заданного правила проектирования. В случае выражения (4) наборы $x_i|1, x_j|0$, $x_i|0, x_j|1$ или $x_i|0, x_j|0$ являются разрешенными. С использованием аналогичных преобразований все множество входных правил проектирования может быть представлено в виде КНФ:

$$F(x_0, x_1, \dots, x_n) = \neg P_0 \wedge \neg P_1 \wedge \dots \wedge \neg P_m. \quad (5)$$

Размер формулы (5) линейно зависит от размера исходного логического высказывания.

2) Симметрии. Исходная задача рассматривает только те правила проектирования, которые пересекают или касаются границы ячеек. Для того чтобы рассмотреть все необходимые правила, нужно построить окно, достаточно большое, чтобы вместить все правила. Высота такого окна фиксирована и зависит от архитектуры, ширина определяется самым широким входным правилом.

В исходной задаче каждая граница ячейки может рассматриваться как ось симметрии (как в горизонтальном, так и в вертикальном направлении). Иными словами, каждому объекту слева от границы можно сопоставить объект справа от нее.

Данное отношение может интерпретироваться следующим образом: если объект x_i слева от границы разрешен, то должны быть разрешены и соответствующие объекты справа и снизу от границ. Если объект x_i запрещен, то все они также запрещены.

Таким образом, входная КНФ формула (5) должна быть расширена дополнительными ограничениями на равенство:

$$eq\ (x_i, x_j) = (\neg x_i \wedge \neg x_j) \vee (x_i \wedge x_j)$$

$$F(x_0, \dots, x_l) = \neg P_0 \wedge \neg P_1 \wedge \dots \wedge \neg P_m \wedge eq\ (x_i, x_j)$$

$$\wedge eq\ (x_k, x_l) \dots \ . \quad (6)$$

Для n объектов топологии к формуле (5) необходимо добавить дизъюнктов.

3) Поиск всех решений. SAT позволяет найти некое решение для формулы (6), которая представляет собой описание всех входных правил проектирования и отношений симметрии между объектами топологии. Каждое решение представляет собой описание множества объектов внутри окна, которое удовлетворяет всем заданным ограничениям. На данном этапе необходимо найти все решения формулы F. Прямолинейный подход требует вызова SAT и сохранения найденного решения:

$$S_0 = SAT(F).$$

Теперь можно взять полученное решение и добавить его отрицание к исходной формуле:

$$S_1 = SAT(F \wedge \neg S_0).$$

Такой запрос можно интерпретировать следующим образом: «найди любое множество значений переменных, удовлетворяющее F, за исключением S_0». Повторяя этот процесс с каждым новым решением, можно получить все решения формулы (6). Более эффективные подходы рассматриваются в работах [16, 17]. Стоит отметить, что SAT относится к классу точных алгоритмов, и данный подход позволяет найти все возможные решения с абсолютной точностью.

Результатом работы описанного алгоритма (англ. AllSat) является формула в дизъюнктивной нормальной форме (ДНФ), каждая конъюнкция в которой соответствует удовлетворяющему набору значений всех переменных F или же легальной топологии в заданном окне (рис. 2):

$$LL\ (x_0, x_1, x_2, x_3, x_4) = \neg x_0 \neg x_1 \neg x_2 \neg x_3 \neg x_4 \vee$$

$$\vee \neg x_0 x_1 \neg x_2 \neg x_3 \neg x_4 \vee \neg x_0 \neg x_1 x_2 \neg x_3 \neg x_4 \vee$$

$$\vee \neg x_0 \neg x_1 \neg x_2 x_3 \neg x_4 \vee \neg x_0 x_1 \neg x_2 x_3 \neg x_4. \quad (7)$$

О каждом конъюнкте на данном этапе известно только то, что соответствующая топология не нарушает никаких правил проектирования, сопоставленная сама с собой (рис. 3, а).

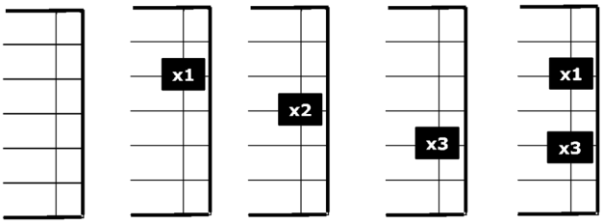

Рис. 2. Примеры топологий, найденных AllSAT

В то же время некоторые пары решений производят нарушения правил проектирования (рис. 3, б).

На следующем этапе необходимо найти такие группы решений, в которых любая пара топологий не производит нарушений правил проектирования.

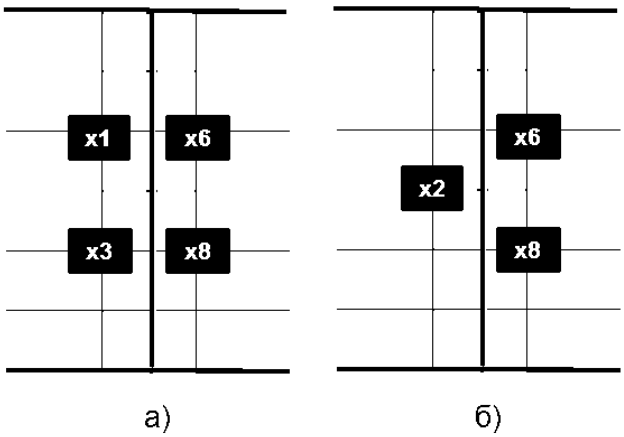

Рис. 3. Примеры сопоставленных топологий

C. Поиск групп решений

На данном этапе необходимо сгруппировать конъюнкты из ДНФ, полученной при помощи AllSAT алгоритма. Каждый конъюнкт соответствует топологии в заданном окне. При этом любая пара топологий в пределах одной группы не производит нарушений правил проектирования. Парные отношения между объектами удобно описывать **кликами**.

Пусть $G = \{E, V\}$ — неориентированный граф, где E — множество ребер, V — множество вершин. Кликой называют полный подграф в G. Иными словами, все вершины в клике C соединены ребрами. Таким образом, проблема поиска групп топологий может быть сведена к перечислению клик графа G. Особый интерес при этом представляют максимальные клики — полные подграфы, которые не могут быть расширены дополнительными вершинами. Такие клики включают в себя все совместимые между собой топологии и обеспечивают максимум ресурсов для трассировщика.

Задача поиска максимальных клик широко известна и для ее решения представлено несколько различных алгоритмов [18-21]. Проблема поиска клик NP-сложная и может потребовать экспоненциально много времени и памяти при решении.

Пусть вершины v_i и v_j соответствуют конъюнктам с индексами i и j в формуле (7). Ребро e_{ij} соединяет эти две вершины тогда и только тогда, когда вместе они не производят нарушений правил проектирования. Для построения соответствующего графа необходимо $O(n^2)$ операций сопоставления пар топологий.

Таким образом, поиск классов топологий, описываемых формулой (7), сводится к поиску максимальных клик графа G. Для перечисления клик в данной работе используется алгоритм, представленный в работе [21].

Результатом на данном этапе является список клик графа G (рис. 4). В рассматриваемом примере граф G (рис. 4, а) содержит две максимальные клики (рис. 4, б, в). Каждая клика соответствует набору конъюнкций в ДНФ формуле (7).

На следующем этапе необходимо построить компактное представление каждой клики в форме булевых выражений. Полученная формула должна запрещать все возможные топологии, за исключением перечисленных в заданной клике.

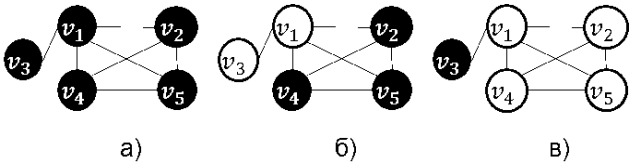

Рис. 4. Примеры найденных групп топологий. Белым отмечены вершины максимальных клик

D. Компактное описание правил

В качестве входных данных используется формула (7), перечисляющая все возможные топологии в заданном окне, и множество клик графа G, каждая из которых соответствует определенным конъюнкциям в формуле (7).

Задачей данного этапа является поиск отрицания формулы (7) и поиск его минимального представления. Подобное преобразование можно выполнить с использованием методов логической минимизации.

Задача логической минимизации сводится к поиску эквивалентного описания заданной логической формулы. Дополнительно рассматриваются ограничения, такие как количество логических вентилей, необходимых для построения соответствующего логического устройства, производительность, площадь устройства и пр. Проблема логической минимизации принадлежит к классу $\sum_2^p -$ полных [22]. В работах [23-26] рассматриваются различные подходы решения этой задачи.

На входе мы имеем конъюнкции, для которых формула (7) возвращает значение «истина». Результатом является множество термов, для которых формула (4) возвращает значение «ложь».

Далее мы будем придерживаться следующей нотации: символы ‘0’, ‘1’ и ‘-‘ в i-й позиции совпадают с $x_i, \neg x_i$ и значением «не важно» соответственно. Таким образом, конъюнкция $\neg x_0 x_1 \neg x_3$ для переменных $x_0 \dots x_3$ будет записана как $01-0$.

Рассмотрим пример с двумя соседними объектами, которые не могут присутствовать одновременно. На предыдущем этапе были построены следующие клики:

$$c_1 = \{v_1, v_2, v_4, v_5\},$$
$$c_2 = \{v_1, v_3\}.$$

Пользуясь методами логической минимизации, для каждой формулы, соответствующей клике, было получено компактное описание отрицания:

$$\neg c_1 = x_0 \vee x_2 \vee x_4,$$
$$\neg c_2 = x_0 \vee x_1 \vee x_3 \vee x_4.$$

Таким образом, для формулы (7) и множества клик были построены компактные отрицания в виде КНФ формул.

E. Дальнейший анализ

В результате работы были получены различные варианты вспомогательных правил. Теперь необходимо выбрать среди них вариант, который наилучшим образом соответствует требованиям к результату трассировки стандартных ячеек. Такой вариант может быть выбран специалистом вручную. Однако возможно внедрение дополнительной автоматизации и поддержки с использованием следующего подхода:

1) выбрать несколько наиболее приоритетных стандартных ячеек;
2) оттрассировать все выбранные ячейки с учетом определенных вспомогательных правил;
3) проанализировать параметры каждой ячейки: площадь, задержки, энергопотребление и пр.;
4) сравнить параметры каждого набора правил;
5) выбрать наиболее подходящий набор правил.

III. Результаты

Подход, описанный в разделе II, позволил нам построить несколько вариантов вспомогательных правил с использованием алгоритма AllSAT и перечисления максимальных клик графа. Для проверки результатов были выбраны несколько наборов различных правил проектирования, каждый из которых рассматривает различные объекты топологии.

Табл. 1 представляет результаты экспериментов. Первая и вторая колонки представляют количество переменных и ограничений в AllSAT задаче. Следующие две колонки представляют количество возможных решений для формулы (6) и количество максимальных клик в соответствующем графе. Последние две графы показывают время, затраченное на поиск решения методом AllSAT и алгоритмом перечисления максимальных клик.

Таблица 1

Экспериментальные результаты

Литералы	Дизъюнкции	Решения	Клики	AllSat, с	Поиск клик, с
65	16	13	4	0.006	0.004
348	585	2	2	0.023	<0.001
574	271	1	1	0.09	<0.001
1567	3586	16	16	0.024	0.012
5847	5723	19075	1	0.385	45.134
16164	30996	20136	7798	1.075	143.278

Число решений, полученных алгоритмом AllSAT, оказывает сильное влияние на общую производительность программы. Даже в случае больших задач все возможные топологии находятся в кратчайшие сроки. Однако число полученных решений оказывает влияние на последующие этапы: построение графа, описывающего отношения между топологиями, и поиск максимальных клик. Эти наблюдения соответствуют теоретическим выводам о сложности трансляции формул из КНФ представления в ДНФ.

IV. Заключение

В данной работе был продемонстрирован способ получения вспомогательных правил проектирования, необходимых в синтезе стандартных ячеек.

На первом этапе правила проектирования в форме булевых выражений записаны как КНФ формула. К полученному выражению добавлены также ограничения на симметрию, после чего итоговая формула решена методом AllSAT. В результате были получены все возможные решения для заданной КНФ или, иначе говоря, все возможные топологии, которые не нарушают правила проектирования.

На втором этапе были построены группы топологий, в которых любая пара решений не нарушает правил проектирования в случае, когда топологии размещены рядом друг с другом. Для поиска подобных классов топологий использовалась процедура перечисления максимальных клик графа.

На заключительном этапе искомые правила проектирования были построены с использованием групп топологий. Для их получения были применены методы минимизации логических функций.

Стремительный прогресс в методах решения задач выполнимости булевых функций позволяет работать над проблемами, которые сложно формализовать и решать с использованием явных алгоритмов. Подход, описанный в данной работе, может также применяться в решении задач автоматического планирования, составлении расписаний, криптографии.

В данной работе нам удалось построить метод автоматического получения вспомогательных правил проектирования для синтеза стандартных ячеек. Эти правила позволяют избегать нарушений правил проектирования, возникающих при размещении стандартных ячеек рядом. Ранее для решения этой задачи требовались значительные объемы ручного труда и написание сложных скриптов. Было продемонстрировано, что вспомогательные ограничения на границах могут быть получены напрямую из правил проектирования и описания архитектуры.

В качестве направления для дальнейших исследований необходимо отметить сложность конверсии КНФ формулы в форму ДНФ. Для выполнения такого преобразования может потребоваться экспоненциально много вычислительных ресурсов. Необходимо исследовать возможность уменьшения сложности данного этапа или же исключить его вовсе. Производительность процедуры перечисления максимальных клик графа также может вызвать затруднения в некоторых случаях.

Открытым вопросом является и процедура выбора наиболее подходящего множества ограничений среди нескольких вариантов. Для ряда задач может быть построено слишком много вариантов решений, чтобы их можно было проанализировать вручную или с использованием экспериментальной оценки.

Литература

[1] Chlebík M., Chlebíková J. Approximation hardness of the Steiner tree problem on graphs // Algorithm Theory—SWAT 2002. – Springer Berlin Heidelberg, 2002. – С. 170-179.

[2] Berger B. et al. Nearly optimal algorithms and bounds for multilayer channel routing // Journal of the ACM (JACM). – 1995. – Т. 42. – №. 2. – С. 500-542.

[3] Xu G. et al. Redundant-via enhanced maze routing for yield improvement // Proceedings of the 2005 Asia and South Pacific Design Automation Conference. – ACM, 2005. – С. 1148-1151.

[4] Huang L. D., Wong M. D. F. Optical proximity correction (OPC): friendly maze routing // Proceedings of the 41st annual Design Automation Conference. – ACM, 2004. – С. 186-191.

[5] Iizuka T., Ikeda M., Asada K. High speed layout synthesis for minimum-width CMOS logic cells via Boolean satisfiability // IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences. – 2004. – Т. 87. – №. 12. – С. 3293-3300.

[6] Nam G. J., Sakallah K. A., Rutenbar R. A. Satisfiability-based layout revisited: detailed routing of complex FPGAs via search-based Boolean SAT // Proceedings of the 1999 ACM/SIGDA seventh international symposium on Field programmable gate arrays. – ACM, 1999. – С. 167-175.

[7] Taylor B., Pileggi L. Exact combinatorial optimization methods for physical design of regular logic bricks // Proceedings of the 44th annual Design Automation Conference. – ACM, 2007. – С. 344-349.

[8] Рыженко Н.В., Сорокин А.А., Быков С.А. Синтез блоков памяти с использованием представления правил в виде булевых функций от топологических объектов // Проблемы разработки перспективных микро- и наноэлектронных систем - 2014. Сборник трудов / под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМ РАН, 2014. Часть I. С. 127-132.

[9] Ryzhenko N., Burns S. Standard cell routing via boolean satisfiability // Proceedings of the 49th Annual Design Automation Conference. – ACM, 2012. – С. 603-612.

[10] Рыженко Н.В., Сорокин А.А., Быков С.А., Талалай М.С. Минимизация числа нежелательных топологий при проектировании стандартных ячеек // Проблемы разработки перспективных микро- и наноэлектронных систем - 2014. Сборник трудов / под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМ РАН, 2014. Часть I. С. 121-126.

[11] Cortadella J. et al. A boolean rule-based approach for manufacturability-aware cell routing //Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on. – 2014. – Т. 33. – №. 3. – С. 409-422.

[12] Suto G. Rule agnostic routing by using design fabrics // Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE. – IEEE, 2012. – С. 471-475.

[13] Zhang L., Malik S. The quest for efficient boolean satisfiability solvers // Computer Aided Verification. – Springer Berlin Heidelberg, 2002. – С. 17-36.

[14] Tseitin G. S. On the complexity of derivation in propositional calculus // Automation of reasoning. – Springer Berlin Heidelberg, 1983. – С. 466-483.

[15] Plaisted D. A., Greenbaum S. A structure-preserving clause form translation // Journal of Symbolic Computation. – 1986. – Т. 2. – №. 3. – С. 293-304.

[16] McMillan K. L. Applying SAT methods in unbounded symbolic model checking // Computer Aided Verification. – Springer Berlin Heidelberg, 2002. – С. 250-264.

[17] Yu Y. et al. All-SAT using minimal blocking clauses // VLSI Design and 2014 13th International Conference on Embedded Systems, 2014 27th International Conference on. – IEEE, 2014. – С. 86-91.

[18] Östergård P. R. J. A fast algorithm for the maximum clique problem // Discrete Applied Mathematics. – 2002. – Т. 120. – №. 1. – С. 197-207.

[19] Tomita E. et al. A simple and faster branch-and-bound algorithm for finding a maximum clique // WALCOM: Algorithms and computation. – Springer Berlin Heidelberg, 2010. – С. 191-203.

[20] Pattabiraman B. et al. Fast algorithms for the maximum clique problem on massive sparse graphs // Algorithms and Models for the Web Graph. – Springer International Publishing, 2013. – С. 156-169.

[21] Makino K., Uno T. New algorithms for enumerating all maximal cliques // Algorithm Theory-SWAT 2004. – Springer Berlin Heidelberg, 2004. – С. 260-272.

[22] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness // Freeman. New York, 1979.

[23] Buchfuhrer D., Umans C. The complexity of boolean formula minimization // Journal of Computer and System Sciences. – 2011. – Т. 77. – №. 1. – С. 142-153.

[24] Brayton R. K. et al. Logic minimization algorithms for VLSI synthesis. – Springer Science & Business Media, 1984. – Т. 2.

[25] Sapra S., Theobald M., Clarke E. SAT-based algorithms for logic minimization // Computer Design, 2003. Proceedings. 21st International Conference on. – IEEE, 2003. – С. 510-517.

[25] Hlavička J., Fišer P. BOOM: a heuristic boolean minimizer // Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design. – IEEE Press, 2001. – С. 439-442.

Automatic Determining of Auxiliary Constraints at Boundaries for Standard Cells Synthesis Flow

S.A. Bykov, N.V. Ryzhenko, A.A. Sorokin

AO “Intel A/O”, sergey.bykov@intel.com

Keywords — routing, standard cells, design rules, cell library, optical lithography, Boolean satisfiability.

ABSTRACT

Standard cell methodology significantly reduces the complexity of the design. A library of standard cells is designed in such a way that guarantees that any possible allowed placement of standard cells does not violate any design rule.

Due to the large number and complexity of design rules, it is a difficult task to formulate such additional constraints that they will prevent any DRVs between abutted cells. Moreover, it is difficult to generalize this approach – each new technology or a set of design rules requires new algorithms and approaches to generate several variants of rules and choose the most appropriate one.

The current paper proposes a solution for automatic exploration of additional boundary constraints. Key components are: description of design rules using Boolean expressions over discrete layout objects; usage of Boolean satisfiability framework to generate all possible DR-clean layouts within a clip; maximal clique’s enumeration procedure to divide layouts into groups; logical minimization to compute resulting boundary constraints in the form of Boolean expressions. Experimental results demonstrated applicability of the proposed approach for a wide class of design rules.

At the first stage, we build a clip, big enough to contain input design rules. The clip includes a set of cell boundaries. Then rules are translated into a set of Boolean expressions [1], expressed as conjunctive normal form (CNF). Cell boundaries can be considered as symmetries axis: if we have allowed placing an object on one side from a boundary, we must allow using a corresponding object on the other side. We extend the CNF formula with such equivalence constraints.

At the second stage, we use Boolean satisfiability framework and AllSat techniques [2] to enumerate all possible DR-clean layouts within the given clip. At this point we know that it is safe to place each layout to itself – such pairs do not produce design rules violations.

At the third stage, we use graph representation to find groups of layouts. We build a graph, where each vertex corresponds to a layout. We draw an edge between a pair of layouts if and only if they did not produce DRVs together. Each maximal clique in such a graph corresponds to a set of layouts, which are safe to abut to each other. We enumerate them all using algorithm that is described in [3].

At the last stage, we use logic minimization framework [4]-[5] to compute additional constraints on boundaries using layouts, which correspond to a clique. Thus, each clique produces a specific set of design rules.

The algorithm was used to process several sets of design rules and to generate corresponding additional constraints on cell boundaries. For a small test case it took approx. 1 min to generate 13 layouts and group them into 4 rules variants. The task included 65 Boolean variables and 16 clauses. The largest test case involved 16164 literals and 30996 clauses. 20136 layouts were found, they were grouped into 7798 classes. 8 min were required to process the task.

REFERENCES

[1] Suto G. Rule agnostic routing by using design fabrics // Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE. – IEEE, 2012. – С. 471-475.

[2] Yu Y. et al. All-SAT using minimal blocking clauses // VLSI Design and 2014 13th International Conference on Embedded Systems, 2014 27th International Conference on. – IEEE, 2014. – С. 86-91.

[3] Makino K., Uno T. New algorithms for enumerating all maximal cliques // Algorithm Theory-SWAT 2004. – Springer Berlin Heidelberg, 2004. – С. 260-272.

[4] Buchfuhrer D., Umans C. The complexity of boolean formula minimization // Journal of Computer and System Sciences. – 2011. – Т. 77. – №. 1. – С. 142-153.

[5] Sapra S., Theobald M., Clarke E. SAT-based algorithms for logic minimization // Computer Design, 2003. Proceedings. 21st International Conference on. – IEEE, 2003. – С. 510-517.