Метод минимизации задержек сигналов при сжатии топологии с учетом технологической сетки

к.т.н. Плеханов А.С.

29 ноября 2003 г.

Аннотация

В статье предложен метод минимизации суммарной задержки сигналов путем определения координаты каждого топологического объекта в пределах их допустимых интервалов При этом учитываются не только технологические ограничения но и требования размещения некоторых объектов в координатах, кратных шагу технологической сетки.

Введение

Компакция (сжатие) топологии является важной процедурой в САПР интегральных микросхем. Задачей компакции является не только уменьшение площади, занимаемой схемой, но и улучшение характеристик схемы путем оптимизации длин проводников. Появление новых технологий, а также требований других компонентов САПР приводят к дополнительным ограничениям, например, к требованию размещения некоторых объектов (чаще всего портов) на технологической сетке. Известные в настоящее время подходы к оптимизации задержек сигналов после этапа компакции топологии не могут охватывать ограничения такого рода.

В данной статье предложен метод, позволяющий минимизировать суммарную задержку сигналов в схеме с учетом требований установки отдельных элементов в координаты, кратные шагу технологической сетки.

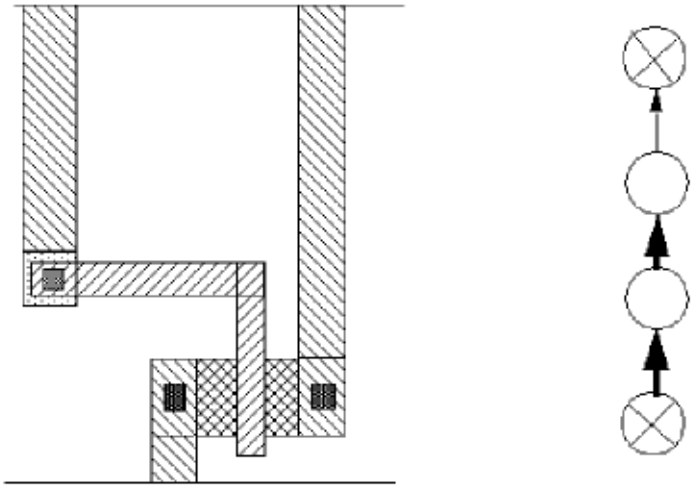

Рис. 1. Пример топологии и графа ограничений

1. Формулировка проблемы

Классический метод одномерной компакции использует граф ограничений. Известно [1], что граф ограничений представляет собой направленный, взвешенный граф $G = (N, E)$, где $N = \{N_i\}$ - множество вершин, $E = \{E_i\}$ - множество ребер. Каждая вершина графа представляет собой некоторый объект топологии и характеризуется его координатой. Ребро описывает множество ограничений между парами объектов и имеет длину L, равную длине максимального ограничения. Такие ограничения имеют вид:

$$X_j - X_k >= L_i, \qquad i = 1 \ldots M, \tag{1}$$

где M - количество ребер, L_i - длина ребра между вершинами, имеющими координаты X_j и X_k [2].

На рисунке 1 показан фрагмент топологии и соответствующий ему граф ограничений. Компакция топологии заключается в определении минимально возможных координат топологических объектов, вычислением минимально возможных координат вершин графа, соответствующих этим вершинам. При таком подходе длина проводников после компакции не является минимальной, что приводит к ухудшению электрических характеристик схемы, в частности, к увеличению задержек. Поэтому после компакции включают этап минимизации суммарной задержки

сигналов, возможной за счет того, что после сжатия топологии многие объекты имеют некоторый диапазон возможных координат. Варьируя координатами объектов в этих диапазонах, можно оптимизировать задержку сигналов. Математически задача сводится к поиску минимума функции вида:

$$\sum_{i=1}^{K} W_i(X_j - X_k), \tag{2}$$

где K - количество проводников, X_j, X_k - координаты вершин графа, соответствующие топологическим объектам, подключенным к данному проводнику, W_i - некоторый параметр, характеризующий данный проводник и зависящий от его ширины и удельного сопротивления. При минимизации данной функции необходимо учитывать ограничения, накладываемые на координаты вершин, выраженные ребрами графа (1). В изложенной выше задаче координаты вершин графа могут принимать любые значения в пределах допустимого интервала. Однако, возникают дополнительные ограничения на координаты некоторых объектов, позволяющие им принимать только ряд дискретных координат в пределах допустимого интервала. Обычно требуется разместить некоторые объекты в координатах, кратных шагу технологической сетки. Математически это требование может быть представлено в виде:

$$X_i = Pn + B, \qquad i = 1 \ldots T, \tag{3}$$

где X_i - координата вершины графа, соответствующего объекту, размещаемому в сетке. T - число объектов, координаты которых должны быть кратны шагу сетки, P - период, B - смещение технологической сетки относительно начала координат, n - целое число. Важно подчеркнуть, что задача минимизации (2) с ограничениями (1) и (3) является нелинейной в отличии от задачи минимизации (2) с ограничениями (1), которая является линейной.

2. Существующий метод решения задачи минимизации суммарной задержки сигналов

Задача минимизации функции (2) с ограничением (1) решается известным методом линейного программирования - симплекс-методом [3]. Однако, этот метод требует обработки большого объема информации в

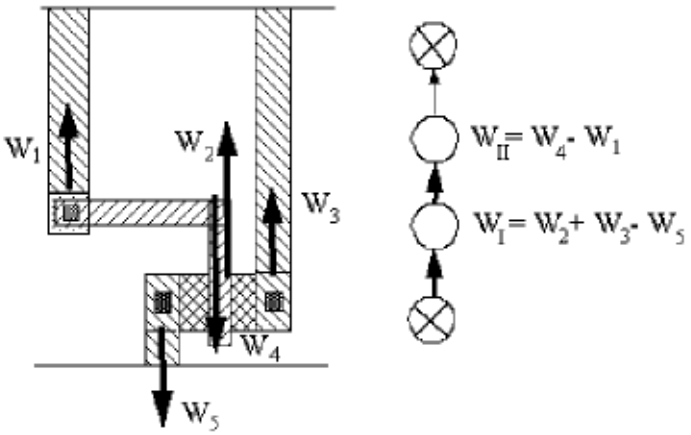

Рис. 2. Пример вычисления веса вершины графа

виде таблиц, поэтому был предложен подход, использующий граф ограничений [4], каждой вершине которого приписывается вес, равный сумме весов W_i проводников, подходящих к данной вершине, взятой со знаком плюс, если направление проводника от вершины совпадает с направлением компакции, и знаком минус - в противном случае (рис. 2). Вычисленный таким образом вес вершины показывает направление перемещения объекта топологии, связанного с этой вершиной, при котором суммарная задержка будет уменьшаться. Если вес вершины больше нуля, то это направление совпадает с направлением компакции, если меньше нуля - это противоположное направление. Симплекс-метод [3] минимизирует функцию переходом от одного допустимого решения к другому, при этом необходимо, чтобы менялась только одна базовая переменная. Это требование приводит к тому, что в графе ограничений в каждый момент времени существует множество подграфов, каждый из которых является деревом. В вершинах этих деревьев находятся неподвижные вершины, то есть вершины, координаты которых нельзя изменять. Каждое ребро, принадлежащее такому подграфу, имеет минимально возможную длину и называется критическим. Такие ребра соответствуют базовым переменным симплекс-метода. Минимизация может начинаться только с допустимого решения. Доказано [4], что граф, соответствующий тополо-

гии, полученной в результате компакции, удовлетворяет такому требованию, так как на нем всегда можно выделить множество соответствующих подграфов, являющихся деревьями.

3. Предложенный алгоритм минимизации без учета требования установки в сетку

В работе [4] излагаются только общие идеи, поэтому был разработан алгоритм, реализующий данный метод и являющийся основой другого алгоритма, учитывающего также ограничения, связанные с технологической сеткой. Опишем первый алгоритм этого метода и будем использовать его для дальнейших ссылок. Процесс нахождения минимума функции (2) на графе ограничений можно представить следующим образом.

Алгоритм 1

Исходные данные: граф ограничений, координаты вершин графа после компакции топологии, веса вершин графа. Результат: координаты вершин графа, минимальная суммарная задержка сигналов.

do {
Repeat Flag = 0
for (цикл по всем критическим ребрам графа) {
Edge (i) - текущее критическое ребро.
1.1. From Node <– вершина, из которой выходит ребро Edge (i).
1.2. To Node <– вершина, к которой идет ребро Edge (i).
1.3. Создаем группу вершин по ребру Edge (i).
1.4. Вычисляем вес группы W.
if ((From Node входит в группу) и (W > 0)) {
2.1. Перемещаем группу по направлению компакции до тех пор, пока это допустимо ребрами графа.
2.2. Находим ребро, препятствующее дальнейшему движению группы.
2.3. Помечаем это ребро как критическое, убираем с ребра Edge (i) пометку критическое.
2.4. Repeat Flag = 1
}
else {
if ((To Node входит в группу) и (W < 0)) {
3.1. Перемещаем группу по направлению,

противоположному направлению компакции до тех пор, пока это допустимо ребрами графа.

3.2. Находим ребро, препятствующее дальнейшему движению группы.

3.3. Помечаем это ребро как критическое, убираем с ребра Edge (i) пометку критическое.

3.4. Repeat Flag = 1

}

}

}

} while (Repead Flag);

Поясним создание группы вершин графа. Пусть имеется критическое ребро графа. Находим вершину, из которой выходит данное ребро *(FromNode)* и вершину, к которой оно идет *(ToNode)*. Находим подграф, в который входит данное ребро. Делим данный подграф таким образом, чтобы получить два подграфа, в один из которых входит *FromNode*, а в другой входит *ToNode*. Поскольку исходный подграф является деревом, очевидно, что полученные подграфы также являются деревьями. Из полученных двух подграфов выбираем тот, который не содержит неподвижной вершины, являющейся вершиной исходного подграфа. Из вершин этого подграфа образуем группу, которая будет использована для последующей обработки.

Проиллюстрируем это на примере. На рисунке 3 показан граф, соответствующий некоторой топологии после компакции. Все вершины графа имеют минимально возможные координаты. Вычислим вес каждой вершины. Возьмем ребро между вершинами N8 *(FromNode)* и N9 *(ToNode)*. Это ребро принадлежит подграфу, в который входят вершины Source, N3, N4, N5, N7, N8, N9 и N10, а принадлежащие ему ребра показаны сплошными стрелками. Делим этот подграф на два подграфа таким образом, чтобы вершина N8 входила в один из них, а N9 - в другой. Первый подграф состоит из вершин Source, N3, N4, N7 и N8, а второй - из вершин N9, N5, N6 и N10. Теперь вершина Source, являющаяся вершиной исходного подграфа, содержится в первом из полученных подграфов. Следовательно, группу формируем из вершин второго из полученных подграфов, то есть из вершин N9, N5, N6 и N10. Вычисляем вес полученной группы как сумму весов всех входящих в нее вершин. Аналогично весу отдельной вершины, вес группы показывает направление ее перемещения для уменьшения суммарной задержки. Цикл *do - while* выполняется до тех пор, пока возможно хотя бы одно изменение координат вершин графа, приводящее к уменьшению целевой функции.

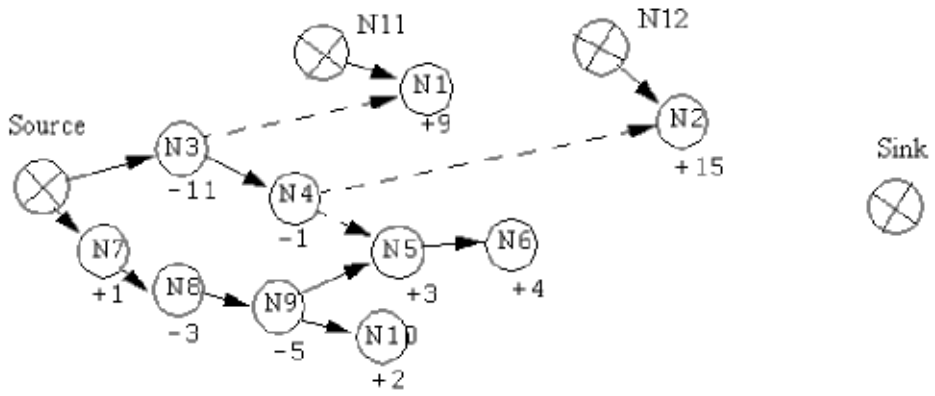

Рис. 3. Определение группы вершин графа по ребру между вершинами N8 и N9

Следует отметить важное обстоятельство: в результате этих действий, несмотря на изменение состава и конфигурации подграфов, общая структура графа в виде множества деревьев остается неизменной.

3.1. Пример алгоритма

Изложенный выше алгоритм поясним на примере. Предположим, что при обходе всех критических ребер графа первым было выбрано ребро между вершинами N8 и N9 (рис.3). Как было показано ранее, группа, образованная этим ребром, состоит из вершин N9, N5, N6 и N10. В данную группу входит *ToNode* (N9). Вес группы равен 4. Поскольку вес группы должен быть меньше нуля, ее обработку прекращаем и переходим к обработке следующего критического ребра. Выберем следующее критическое ребро, например, между вершинами Source и N3. Образованная этим ребром группа состоит из двух вершин (N3 и N4). Поскольку *ToNode* (N3) входит в эту группу и ее вес отрицательный (-12), перемещаем ее в противоположном компакции направлении до тех пор, пока какое-либо ребро не сократится до минимально возможного размера. В данном случае таким ребром является ребро между вершинами N4 и N5. Отмечаем его как критическое, а с обрабатываемого ребра снимаем эту метку. Полученный граф показан на рисунке 4. Аналогично поступаем с ребром между вершинами Source и N7 (рис. 5), после чего обрабатываем ребро между вершинами N11 и N1 (рис. 6) и ребро между вершинами N7

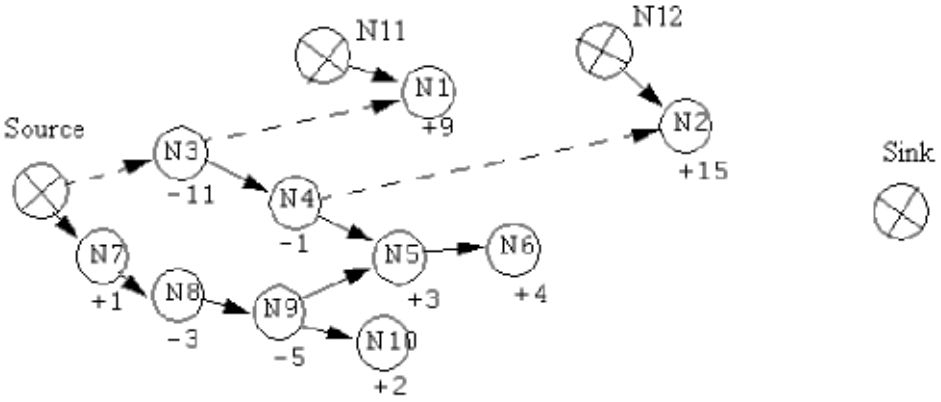

Рис. 4. Перемещение группы, образованной ребром между вершинами Source и N3

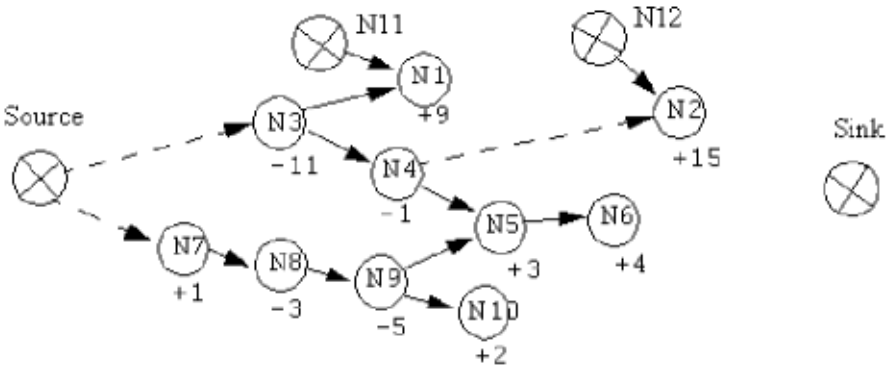

Рис. 5. Перемещение группы, образованной ребром между вершинами Source и N7

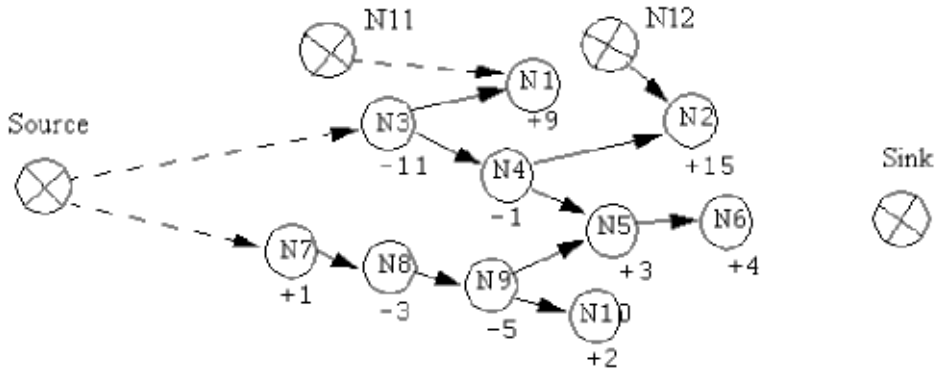

Рис. 6. Перемещение группы, образованной ребром между вершинами N11 и N1

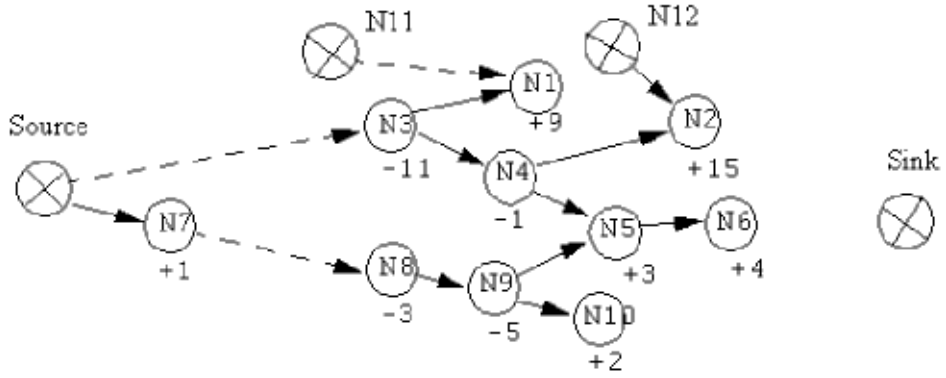

Рис. 7. Оптимальное решение, полученное после перемещения вершины N7

и N8 (рис. 7). Заметим, что в последнем случае в группу входит вершина *FromNode*, а вес группы больше нуля.

4. Проблема существующего метода при обработке объектов в сетке

При учете дополнительных ограничений (3) задача перестает быть линейной. Дополнительное ограничение приводит к тому, что координата объекта топологии, кратная шагу технологической сетки, и координата соответствующей вершины графа могут принимать только дискретные значения на допустимом интервале. В соответствии с этим координаты вершин, входящих в группу, содержащую данную вершину, также будут принимать только дискретные значения. Это приводит к тому, что решение, полученное приведенным выше алгоритмом, будет неоптимальным. Покажем это на примере. На рисунке 8 приведена топология после компакции и соответствующий ей граф. Необходимо найти топологию с минимально возможной суммарной задержкой сигналов и дополнительным условием - координата горизонтального проводника должна быть кратна шагу технологической сетки, показанной пунктирными линиями. В соответствии с приведенным в п.1.3 алгоритма 1, вершины 1 и 2 должны быть объединены в группу. Эта группа и соответствующие ей топологические объекты перемещаются в направлении, противоположном направлению компакции. При этом координата горизонтального проводника должна быть кратна шагу технологической сетки. Результат минимизации приведен на рисунке 9. Очевидно, что существует более оптимальное решение, показанное на рисунке 10, поскольку расстояние между горизонтальным проводником и транзистором больше минимально возможного.

Можно показать, что в общем случае использования алгоритма 1 при дополнительных ограничениях (3) приводит к неоптимальным решениям. Это объясняется тем, что симплекс-метод может быть использован только для решения линейных задач. Точное решение задачи (2) с условиями (3) этим методом требует перебора множества вариантов, что ведет к большим затратам вычислительных ресурсов.

5. Предложенное решение проблемы

Ниже описан метод оптимизации суммарной задержки при дополнительных ограничениях, связанных с технологической сеткой, не требующий перебора вариантов и позволяющий получить оптимальное или

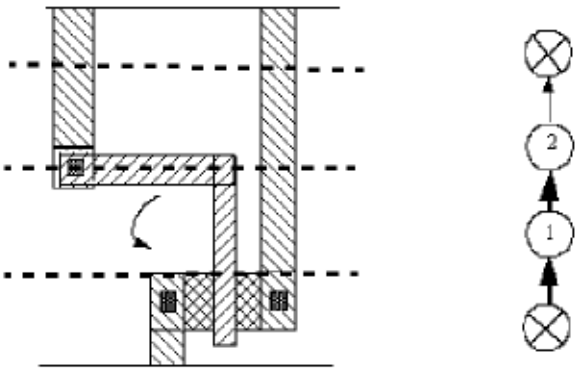

Рис. 8. Топология, полученная после компакции с учетом технологической сетки

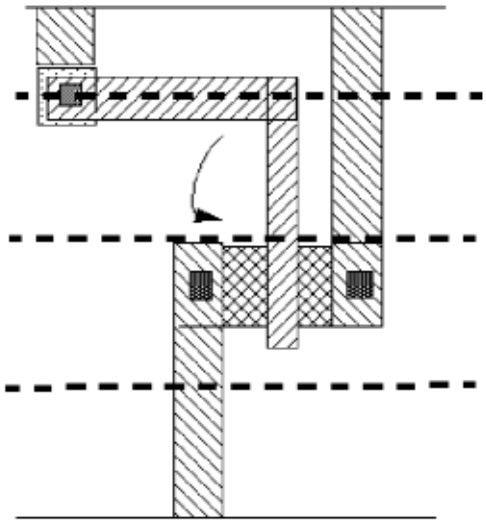

Рис. 9. Неоптимальное решение, полученное при использовании алгоритма 1

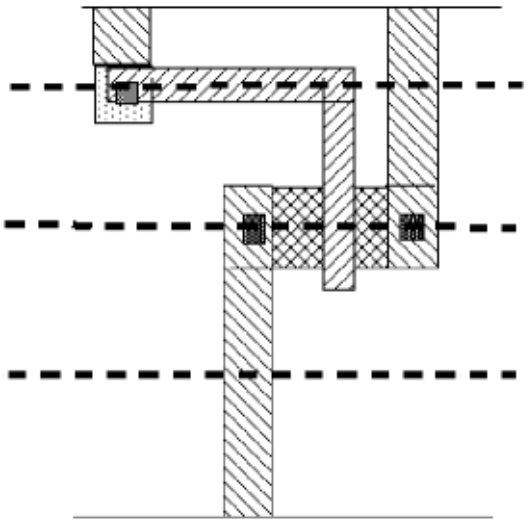

Рис. 10. Оптимальное решение, полученное при использовании алгоритма 1

близкое к оптимальному решение. Метод решения задачи состоит из двух этапов. На первом этапе ограничения, накладываемые сеткой, не учитываются, и решение получают изложенным выше способом. На втором этапе заданные топологические объекты размещаются в координатах, кратных технологической сетки. При этом используется решение, полученное на первом этапе.

Рассмотрим второй этап решения задачи. Поскольку решение, полученное на первом этапе, является оптимальным (без учета требований (3)), то любое изменение координат топологических объектов либо ухудшает значение целевой функции, либо оставляет его неизменным. Для решения задачи (2) с дополнительными условиями (3) необходимо найти такие новые координаты объектов, при которых заданные топологические объекты находились бы в технологической сетке, а ухудшение целевой функции было бы минимальным. Предположим, что имеется один объект, который необходимо разместить в технологической сетке. Для этого необходимо выбрать координату сетки, в которую будет устанавливаться заданный объект топологии. Пусть координатой объекта является X, а ближайшие координаты технологической сетки - X_0 и X_1 ($X_0 < X$, $X_1 > X$) (рис. 11). Для выбора из этих двух координат той, в которую будет установлен объект, используем следующий алгоритм:

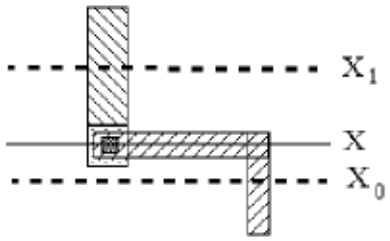

Рис. 11. Выбор координаты технологической сетки для размещения проводника

Алгоритм 2

Исходные данные: граф ограничений, объект топологии, который необходимо установить в координату X_0 или X_1.

Результат: координата X_0 или X_1.

Проверяем условия (1) при установке объекта в координату X_0.
Если условия выполняются, ставим Flag0 = 1, иначе
ставим Flag0 = 0.

Проверяем условия (1) при установке объекта в координату X_1.
Если условия выполняются, ставим Flag1 = 1, иначе
ставим Flag1 = 0.

```
if( ( Flag0 == 1) && ( Flag1 == 0) ) {
  выбираем X0.
}
else {
  if( ( Flag0 == 0 && ( Flag1 == 1 ) ) {
    выбираем X1.
  }
```

else {
if((Flag0 == 1) && (Flag1 == 1)) {
if (X - X_0 < X_1 - X) {
выбираем X_0.
}
else {
выбираем X_1.
}
}
else {
Ошибка.
}
}
}

Таким образом, задача установки объекта в сетку сводится к задаче оптимального перемещения объекта из одной координаты в другую. Ниже приведен алгоритм решения данной задачи, если выбрана координата сетки X_0:

Алгоритм 3

Исходные данные: граф ограничений, вершина графа *pNode* с координатой $X > X_0$, координата X_0, в которую необходимо установить *pNode*.

Результат: координата вершины *pNode* равна X_0, координаты вершин графа, минимальная суммарная задержка сигналов.

RepeatFlag = 1;
do {
1.1. Находят путь от pNode до неподвижной вершины - корня дерева, в которое входит pNode.
1.2. Из всех ребер, находящихся на этом пути, выбирают те, которые имеют направление, совпадающее с направлением этого пути.
1.3. for (множество выбранных ребер) {
Edge (i) - текущее ребро из этого множества.
1.3.1. Создают группу вершин по данному ребру.
1.3.2. Вычисляют вес группы W.
}
1.4. Находят группу с максимальным весом W.
1.5. Перемещают группу по направлению компакции.

1.6. if (pNode достигла заданной координаты) {
1.6.1. Убирают с ребра pEdge (i) пометку "критическое".
1.6.2. Помечают вершину pNode как неподвижную.
1.6.3. RepeatFlag = 0;
}
1.7. else {
1.7.1. Находят ребро, препятствующее дальнейшему движению группы.
1.7.2. Помечают это ребро как критическое, а с ребра pEdge (i) убирают пометку "критическое".
}
} while(RepeatFlag);

Поясним данный алгоритм. Цикл *do* выполняется до тех пор, пока вершина графа и соответствующий ей объект топологии не достигнет заданной координаты. На каждой итерации этого цикла определяется группа вершин, содержащая обрабатываемую вершину *pNode*, такая, что при ее перемещении ухудшение целевой функции было бы минимальным. Для определения такой группы находится путь от вершины *pNode* до вершины, являющейся корнем дерева, в которое входит *pNode* (1.1). Из всех ребер, принадлежащих пути, выбираются те, которые имеют направление, совпадающее с направлением этого пути (1.2). Для каждого такого ребра создаем группу вершин так же, как это делалось в п.1.3 алгоритма 1. Из всех созданных таким образом групп выбираем одну, имеющую максимальный вес (1.4). Перемещаем эту группу вершин по направлению компакции до тех пор, пока вершина *pNode* не достигнет заданной координаты (1.6) или дальнейшему перемещению группы будут препятствовать ребра графа. В первом случае вершина *pNode* помечается как неподвижная, и она становится вершиной дерева, в которое входят вершины, принадлежащие обрабатываемой группе. На этом алгоритм заканчивается. Во втором случае помечаем ребро, препятствующее дальнейшему движению группы, как критическое и переходим на следующую итерацию цикла *do*. Заметим, что структура графа в виде множества деревьев остается неизменной. Если объект топологии необходимо переместить в координату X_1, в данном алгоритме необходимо в пути от вершины *pNode* выбирать ребра, направление которых противоположно направлению пути, а среди полученных групп выбирать группу с наименьшим весом.

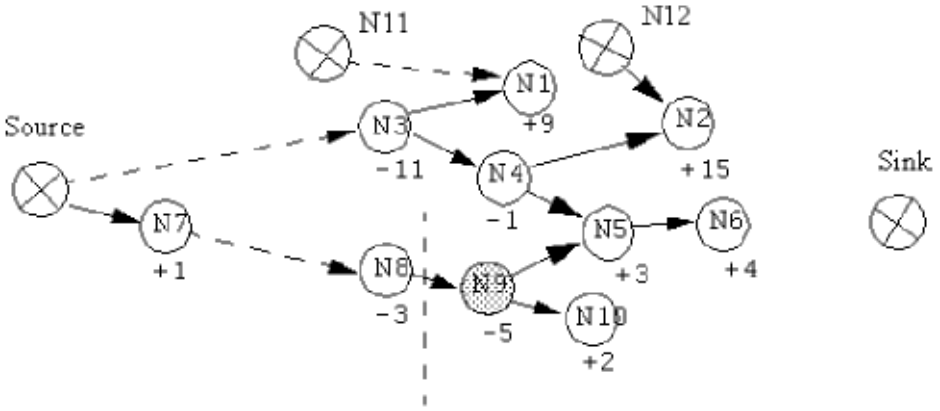

Рис. 12. Выбор группы вершин для обработки

5.1. Пример алгоритма 3

Приведем пример применения алгоритма 3, основываясь на результатах, полученных в примере работы алгоритма 1. На рисунке 12 показан граф, соответствующий топологии, полученной после минимизации суммарной задержки. Предположим, необходимо установить объект, которому соответствует вершина N9, в координату, кратную шагу технологической сетки. Это соответствует преобразованию координаты вершины N9, в координату, условно показанную вертикальной пунктирной линией. Вершина N9 принадлежит к подграфу, являющемуся деревом с корнем в вершине N12. Определим путь от вершины N9 до вершины N12 и найдем ребра, принадлежащие этому пути и имеющие направление, совпадающее с ним. Это ребро между вершинами N9 и N5 и между N4 и N2. По этим ребрам находим группы вершин графа. Группа, образованная ребром между вершинами N9, N5, состоит из вершин с N8, N9 и N10 и имеет вес, равный -6. Группа, образованная ребром между вершинами N4, N2, состоит из вершин N3, N1, N4, N5, N6, N8, N9, N10 и имеет вес, равный -2. Выбираем последнюю группу, поскольку она имеет максимальный вес. Перемещаем ее по направлению компакции до тех пор, пока ребро между вершинами N11 и N1 не сократится до минимального размера (рис. 13), что препятствует достижению вершины N9 требуемой координаты. В полученном новом дереве с корнем в вершине N11 аналогично находим группы вершин, максимальным весом из которых обладает группа, состоящая из вершин N8, N9 и N10 (рис. 14). При

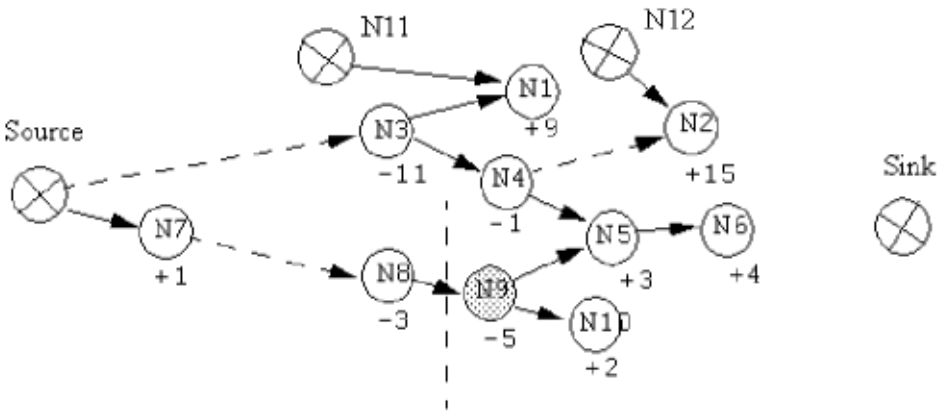

Рис. 13. Перемещение группы, образованной ребром между вершинами N4 и N2

перемещении данной группы вершина N9 достигает заданной координаты (рис. 15).

5.2. Дополнительные замечания

Можно доказать, что перемещение одного топологического объекта в заданную координату (алгоритм 3) дает оптимальное решение. Тот же результат можно получить, если бы после минимизации суммарной задержки сигналов принудительно установить объект в заданную координату и снова выполнить одномерную компакцию топологии и процедуру минимизации задержки. Однако это потребует больших вычислительных ресурсов.

В случае нескольких объектов, координаты которых должны быть кратны шагу технологической сетки, оптимальный результат получается в случае, если такие объекты не влияют друг на друга при применении алгоритма 3. Однако, как показывает практика, даже в случае такого влияния полученный результат близок к оптимальному.

Заключение

В результате изложенного выше можно сделать следующие выводы:

1) Предложен метод минимизации суммарной задержки сигналов с учетом установки заданных топологических объектов в координа-

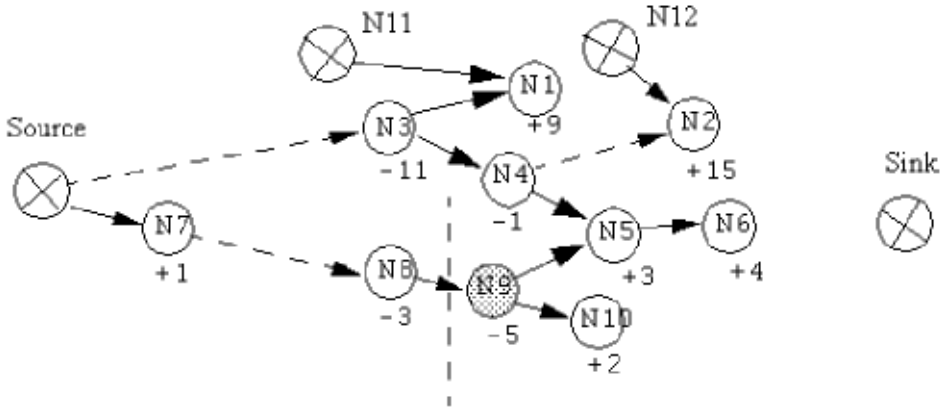

Рис. 14. Перемещение группы, образованной ребром между вершинами N9 и N5

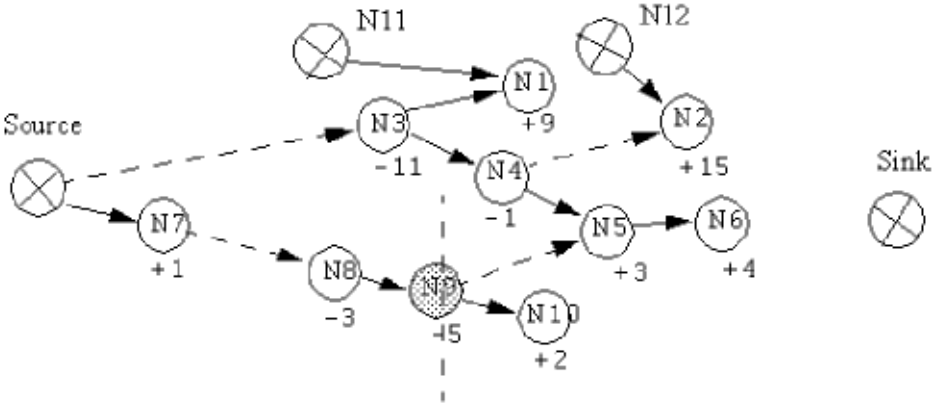

Рис. 15. Структура графа после завершения алгоритма 3

ты, кратные шагу технологической сетки, который позволил находить оптимальные или близкие к оптимальным проектные решения.

2) Метод обладает небольшой дополнительной сложностью по сравнению с известным методом минимизации без дополнительных условий.

3) Разработанный алгоритм, реализующий предложенный метод, использует те же или подобные представления данных и процедуры в процессе вычислений, что и известный алгоритм без дополнительных ограничений, поэтому делает его близким последнему алгоритму и, следовательно, легко реализуемым.

Список литературы

[1] N.Sherwani. "Algorithm for VLSI Physical Design Automation", Second Edition, Kluwer Academic Publishers, 1995, 423pp.

[2] Chi-Yuan Lo, Ravi Varadarajan "An $O(n\log(n))$ 1-d Compaction Algorithm"27th ACM/IEEE Design Automation Conference, 1990, pp. 382-387.

[3] В.В.Лесин Ю.П.Лисовец. "Основы методов оптимизации."Москва, изд. МАИ, 1995 г. с 154-186.

[4] Sching L.Lin Jonathan Allen. "Minplex - A compactor that minimizes the boundary rectangle and individual rectangles in a layout."// Proceeding of the 23rd Design Automation Conference. 1986, pp 123 - 130.

Содержание