1. Стандартная библиотека шаблонов. Алгоритмы. Примеры работы со стандартными алгоритмами)

Набор алгоритмов содержит типовые процедуры, применяемые к контейнерам (поиск, сортировка, слияние). Эти процедуры не являются методами контейнеров классов, а являются шаблонами глобальных функций.

Взаимодействие алгоритмов и контейнеров осуществляется через использование **итераторов,** над которыми можно осуществлять некоторые арифметические операции. Они предостовляют доступ к элементам контейнера.

**Алгоритмы — это самая объёмная и самая востребованная часть библиотеки.**

**1.for\_each()** Наиболее часто используемый алгоритм , **выполнение действия для группы элементов (возможно всех) контейнера**.

**for\_each — алгоритм обхода по умолчанию**

**2.find()** **это поиск некоторого элемента в коллекции**. Многие контейнеры имеют метод find(), который для объекта будет вызываться как obj.find(…), в то время как алгоритм будет вызываться как find( obj:iteator, … ).

Это целая обширная их группа, которую можно объединить по признаку того, что они отбирают элементы коллекции по какому-то признаку, условию, предикату: *find(), find\_if(), find\_if\_not(), find\_first\_of(), find\_end(),* . В эту же группу, можно отнести и *count(), count\_if(), search(), binary\_search(), min(), max(), minmax\_element(), min\_element(), max\_element(), equal()* и др.

**3.Ещё одна условная группа** — это алгоритмы, некоторым образом «тасующие» коллекцию, **переставляющие** **элементы** местами, меняющие значения: **fill()**, **replace\_copy()**, **reverse()**, **rotate()**, **rotate\_copy()**, **shuffle()**, **random\_shuffle()**, **transform()**, **replace()**, **replace\_if()** и др.

**4.** Ещё группа — это алгоритмы **работающие с 2-мя** коллекциями, **копирующие и перемещающие** содержимое (причём, возможно, между коллекциями разного вида, например, **vector<>** в **set<>**): **copy()**, **copy\_if()**, **move()**, **swap\_ranges()**, **remove\_copy()**, **remove\_copy\_if()**, **merge()**,**set\_intersection()**, **set\_difference()** и др.

**5.** И, наконец, совсем особая группа алгоритмов связана с разнообразными **сортировками** **элементов внутри коллекции:** **sort()**, **stable\_sort()**, **is\_sorted()**, **is\_sorted\_until()** и др. Эту интересную группу мы отложим на потом, для отдельного обстоятельного рассмотрения.

**ПРИМЕР**

вопросов, все они работают **подобно** один другому):

|  |  |
| --- | --- |
|  | #include <iostream>  #include <cstring>  #include <vector>  #include <map>  #include <set>  #include <algorithm>  using namespace std;   //перегрузка на ввод символьной строки  inline ostream& operator <<( ostream& out, const vector<char>& obj ) {     for( auto p: obj ) cout << p;     return out;  }    int main( void ) { //есть длинное название книги     char s[] = "The Life and Strange Surprizing Adventures of Robinson \ Crusoe, Of York, Mariner: Who lived Eight and Twenty \  Years, all alone in an un-inhabited Island on the \                 Coast of America, near the Mouth of the Great River of \ Oroonoque; Having been cast on Shore by Shipwreck, \                 wherein all the Men perished but himself. With \                 An Account how he was at last as strangely deliver'd \                 by Pyrates";  // название книги     vector<string> vs = {         "Supercalifragilisticexpialidocious",         // cамое длинное слово на английском языке         "Pneumonoultramicroscopicsilicovolcanoconiosis"         // самый длинный термин на английском языке     };     // copy & find :     vector<char> v1( strlen( s ) ); //длиной название книги     copy( s, s + v1.size(), v1.begin() );     int nb = 0;  //**поиск пробелов в фразе**     for( auto is = find( v1.begin(), v1.end(), ' ' ); is != v1.end();               is = find( ++is, v1.end(), ' ' ) ) nb++;     cout << "в фразе пробелов " << nb << " (" << nb + 1          << " слов)" << endl;  //**поиск миниммов и максимумов**     // min & max :     auto mm = minmax\_element( v1.begin(), v1.end() );     cout << "диапазон символов: '" << \*mm.first << "' ... '"          << \*mm.second << "'" << endl;  //**заполнение и рверс**     // fill & reverse & rotate & shuffle :     vector<char> suv( vs[ 0 ].size() );     copy( vs[ 0 ].begin(), vs[ 0 ].end(), suv.begin() );     cout << suv << endl;     random\_shuffle( suv.begin(), suv.end() );     cout << suv << endl;  //**обратный порядок**     reverse( suv.begin(), suv.end() );     cout << suv << endl;     rotate( suv.begin(), suv.begin() + suv.size() / 2, suv.end() );     cout << suv << endl;     // set\_intersection & set\_difference     set< char > sus, pns;     for( char s: vector<char>( vs[ 0 ].begin(), vs[ 0 ].end() ) )        sus.insert( s );     for( char s: vector<char>( vs[ 1 ].begin(), vs[ 1 ].end() ) )        pns.insert( s );     vector<char> outi( 100 ), outd( 100 );     auto ret = set\_intersection( sus.begin(), sus.end(), pns.begin(),                                  pns.end(), outi.begin() );     cout << "общих литер " << ( ret - outi.begin() ) << " : "          << outi << endl;     ret = set\_difference( sus.begin(), sus.end(), pns.begin(),                           pns.end(), outd.begin() );     cout << "уникальных литер " << ( ret - outd.begin() ) << " : "          << outd << endl;  } |

Здесь использованы контейнеры для **char** (как компактные, но самые неприятные в работе), над которыми выполняются разнообразные алгоритмы практически всех обозначенных групп:

**ПРИМЕР**

|  |  |
| --- | --- |
|  | #include <iostream>  #include <vector>  #include <algorithm>   using namespace std;    inline ostream& operator <<( ostream& out, const vector< unsigned > & obj )  {     cout << "< ";     for( auto& p: obj )        cout << p << " ";     return out << ">";  }    void pow2( unsigned& i ) { i \*= i; }    int main( void ) {     const int examples = 4;     for( int i = 0; i < examples; i++ ) {        unsigned ai[] = { 1, 2, 3, 4 , 5, 6, 7, 8, 9 },             ni = sizeof( ai ) / sizeof( ai[ 0 ] );        vector< unsigned > vi( ai, ai + ni );        cout << vi;        switch( i ) {           case 0:              for\_each( vi.begin(), vi.end(), pow2 );              cout << " => " << vi << endl;              break;           case 1:              for\_each( ai, ai + ni, pow2 );              cout << " => " << vector< unsigned >( ai, ai + ni ) << endl;              break;           case 2:              for( auto& i : ai ) pow2( i );              cout << " => " << vector< unsigned >( ai, ai + ni ) << endl;              break;           case 3:              for\_each( vi.begin() + 2, vi.end() - 2, pow2 );              cout << " => " << vi << endl;              break;        }     }  } |

9, 10. Стандартная библиотека шаблонов. Последовательные контейнеры. Примеры vector.

**Последовательные контейнеры** (или ещё «**контейнеры последовательности**») — это контейнерные классы, элементы которых находятся в последовательности. Их определяющей характеристикой является то, что вы можете вставить свой элемент в любое место контейнера. Наиболее распространённым примером последовательного контейнера является [**массив**](https://ravesli.com/urok-74-massivy-chast-1/): при вставке 4-ёх элементов в массив, эти элементы будут находиться (в массиве) в точно таком же порядке, в котором вы их вставляли.

[**std::vector**](https://ravesli.com/urok-95-vvedenie-v-std-vector-vektory/); Класс**vector**(или просто «**вектор**») — это [**динамический массив**](https://ravesli.com/urok-86-dinamicheskie-massivy/), способный увеличиваться по мере необходимости для содержания всех своих элементов. Класс vector обеспечивает произвольный доступ к своим элементам через **оператор индексации** **[]**, а также поддерживает вставку и удаление элементов.

[**std::array**](https://ravesli.com/urok-94-vvedenie-v-std-array/);

   std::list; **List** (или просто «**список**») — это двусвязный список, каждый элемент которого содержит 2 указателя: один указывает на следующий элемент списка, а второй — на предыдущий элемент списка. list предоставляет доступ только к началу и к концу списка — произвольный доступ запрещён. Если вы хотите найти значение где-то в середине, то вы должны начать с одного конца и перебирать каждый элемент списка до тех пор, пока не найдёте то, что ищите. Преимуществом двусвязного списка является то, что вставка элементов происходит очень быстро, если вы, конечно, знаете, куда хотите вставлять. Обычно для перебора элементов двусвязного списка используются итераторы.

Примеры в другом доке.

**2 ЧАСТЬ**

1. Технологии разработки печатных плат (однослойные, двухсторонние, МПП).

Печатные платы - это элементы конструкции, которые состоят из плоских проводников в виде участков металлизированного покрытия, размещенных на диэлектрическом основании и обеспечивающих соединение элементов электрической цепи. Они **получили широкое распространение в производстве модулей**, ячеек и блоков РЭА **благодаря следующим преимуществам по сравнению** с традиционным монтажом проводниками и кабелями:

* Увеличение плотности монтажных соединений и возможность микроминиатюризации изделий;
* гарантированная стабильность и повторяемость электрических характеристик (проводимости, паразитных емкости и индуктивности);
* повышенная стойкость к климатическим и механическим воздействиям;
* унификация и стандартизация конструктивных и технологических решений;
* увеличение надежности;
* возможность комплексной автоматизации монтажно-сборочных и контрольно-регулировочных работ;
* снижение трудоемкости, материалоемкости и себестоимости

К недостаткам следует отнести сложность внесения изменений в конструкцию и ограниченную ремонтопригодность.

ВИДЫ:

* Односторонние печатные платы (ОПП)
* Двусторонние (ДПП)
* Многослойные (МПП) платы.

Элементами ПП являются:

* диэлектрическое основание,
* металлическое покрытие в виде рисунка печатных проводников и контактных площадок
* монтажные и фиксирующие отверстия.
* контактные площадки-участки металлического покрытия, которые соединяют печатные проводники с металлизацией монтажных отверстий.

ОДНОСТОРОННИЕ ПП:

В этих платах за один технологический цикл получается вся конструкция с монтажными отверстиями и специальными углублениями для расположения ЭРЭ, монтируемых на поверхность.

ДВУСТОРОННИЕ ПП:

Электрическая связь слоев печатного монтажа осуществляется с помощью металлизации отверстий. Двусторонние ПП обладают повышенной плотностью монтажа и надежностью соединений. Они используются в измерительной технике, системах управления и автоматического регулирования..

МНОГОСЛОЙНЫЕ ПП:

Состоят из чередующихся слоев изоляционного материала и проводящего рисунка, соединенных клеевыми прокладками в монолитную структуру путем прессования. Электрическая связь между проводящими слоями выполняется специальными объемными деталями, печатными элементами или химико-гальванической металлизацией. По сравнению с ОПП и ДПП они характеризуются повышенной надежностью и плотностью монтажа, устойчивостью к механическим и климатическим воздействиям, уменьшением размеров и числа контактов.

Методы изготовления ПП

* Субтрактивный
* Аддитивный

В *субтрактивных методах* в качестве основания для печатного монтажа используют фольгированные диэлектрики, на которых формируется проводящий рисунок путем удаления фольги с непроводящих участков.

Аддитивные - Основаны на избирательном осаждении токопроводящего покрытия на диэлектрическое основание, на которое предварительно может наноситься слой клеевой композиции.

Создание рисунка печатного монтажа

следующие методы:

* офсетная печать,
* сеткография,
* фотопечать.

Выбор метода определяется :конструкцией ПП, требуемой точностью и плотностью монтажа, производительностью оборудования и экономичностью процесса.

1. Модели схем в конструкторском проектировании. Модели элементов, модели цепей. Примеры.

С точки зрения конструктора есть платы 1-го уровня и на них устанавливаются микросхемы. Далее платы 1-го уровня устанавливают на платы второго уровня, а все они соединяются на платах третьего уровня. На основании данных по специальной программе из библиотеки элементов подбираются корпуса микросхем, определяется их оптимальное размещение на платах первого уровня, далее оптимально размещают платы первого уровня на платах второго уровня и так далее. Зная расположение ИС и схему их соединений, ЭВМ формирует монтажные схемы и выдает их в виде чертежей. Основные задачи конструкторского проектирования:

* Компоновка - это назначение элементов в корпуса и назначение корпусов на платы.
* Размещение – это распределение элементов одного уровня иерархии на поверхности модуля(платы) следующего уровня иерархии таким образом, чтобы обеспечить максимальное проведение соединений.
* Трассировка – это проведение соединений по заданной схеме соединений с соблюдением заданных технологических ограничений
* Формирование технической документации или носителя для станка с ЧПУ реализации печатных соединений.

Практически все САПР печатных плат содержат:

* Графический редактор, который позволяет нарисовать схему на экране дисплея и перевести ее в модель для представления в памяти ЭВМ .Графические редакторы позволяют поворачивать элементы, создавать собственные начертания элементов
* . Библиотеки элементов, содержащие обширную информацию о компонентах электронных схем от дискретных и электромеханических деталей до существующих и заказных микросборках интегральных схем. Например в пакете САПР ORCAD в библиотеке имеется более 2700 изображений компонентов. пакеты часто имеют выходы на подсистемы моделирования и анализа.

Часто в системы проектирования входят экспертные системы.

**Основные различия** современных САПР печатных плат сводятся к следующему:

1. Процент автоматической трассировки платы.
2. Размер обрабатываемой схемы, то есть число элементов и цепей.
3. Критерии качества, учитываемые при размещении, трассировке, компоновке.
4. Библиотеки элементов (объем, пополнение, стандарты)
5. Возможности для интеграции с другими системами
6. Ввод схем, удобство редактирования и фрагментирования
7. Устройства , для которых система формирует файлы-результаты (принтер, плоттер, фотоплоттер, координатограф, ЧПУ)

Модели схемы

Модель схемы традиционно граф, но вид графа ориентированный он или нет, обычный или мультиграф (граф, где пара вершин соединяется более чем одним ребром) , полный граф (граф, где каждая вершина соединена со всеми остальными) или гиперграф (граф, в котором одно ребро содержит множество вершин) – все это связано с решаемой задачей. Модели схем для разных задач конструирования должны удовлетворять разным требованиям. Универсальная модель утяжеляет алгоритмы автоматизации. Обычно отдельно рассматривают модели элементов и модели цепей.

**Модели элементов**

Традиционно в качестве модели элемента рассматривают вершину графа, если необходимо учитывать контакты элементов, то для двухконтактных элементов это отрезок, соединяющий две вершины. Для многополюсных элементов, где контактов много, моделью элемента является звезда, в центре звезды – вершина элемент, по периферии от центра к вершинам - контактам элемента проводятся отрезки лучи. Иногда используется более жесткая конструкция для многополюсных элементов – многоугольник – число вершин равно числу контактов элемента, в центре многоугольника сам элемент, который соединен ребром с каждой вершиной-контактом элемента.

**Модели цепей**

Часто – это ребро, если есть соединение между элементами ребро есть, ребро отсутствует, когда нет соединения. Такая модель не учитывает, что два элемента могут быть связаны по разным контактам разными электрическими цепями, чтобы это учесть вводили смежные ребра, то есть моделью схемы являлся мультиграф. Однако, мультиграф не учитывал специфику цепи, объединяющей несколько контактов, здесь вообще контакты не учитывались.

Рассмотрим фрагмент схемы из 6 элементов и 9 цепей. Рис.1

I

2

3

4

5

6

1

1

6

5

4

3

2

1

II

III

VII

IX

IV

VIII

VI V

Здесь, каждому элементу схемы соответствует вершина мультиграфа, если в схеме элементы электрически соединены, то в мультиграфе проводится ребро между соответствующими вершинами. Например, элементы 1 и2 в схеме соединены двумя цепями первой I и II в мультиграфе они соединяются двумя ребрами. Второй элемент связан с элементом 3 и 4 цепью IX, с элементами 5 и 6 цепью IV и с элементом 4 цепью III.Кроме того, с 1 элементом цепями I,II.

Представим этот фрагмент схемы **мультиграфом** , как на рис.2

1 2

3

4

5

6

Рис.2 Модель фрагмента схемы – мультиграф.

В **мультиграфе** отсутствуют контакты элементов, входящих в цепи. Если учитывать контакты элементов принадлежащих цепи кроме вершин - элементов, то цепь будет изображаться полным графом, так как все контакты одной электрической цепи связаны электрически друг с другом. Если учитывается направление сигнала, то цепь иногда представляют деревом. Если есть отличие входных и выходных сигналов, то есть учитывают направление движения сигнала, то моделью цепи становится дерево с ориентированными дугами.

Другой моделью схемы является **гиперграф**. Обычный граф схемы - это бинарное отношение. Ребро в нем связывает пару вершин. Гиперграф – n- арные отношения. Гиперребро – это подмножество множества вершин.

Удобный способ представления графов в памяти компьютера с помощью матриц, так для мультиграфа подходит матрица смежности, а для гиперграфа – матрица инцидентности. На рис.3 представлена матрица смежности мультиграфа, который мы видим на рис.2 , для фрагмента схемы рис.1.Здесь на пересечении строки и столбца-число ребер между соответствующими эл-тами

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Номер эл.** | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 0 | 2 | 1 | 0 | 0 | 0 |
| **2** | 2 | 0 | 1 | 2 | 1 | 1 |
| **3** | 1 | 1 | 0 | 1 | 0 | 1 |
| **4** | 0 | 2 | 1 | 0 | 0 | 1 |
| **5** | 0 | 1 | 0 | 0 | 0 | 2 |
| **6** | 0 | 1 | 1 | 1 | 2 | 0 |

Рис.3 Матрица смежности мультиграфа

На рис.4 представлена матрица инцидентности гиперграфа, представленного на рис.1 фрагмента схемы

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| .**Цепи** | **элементы** | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |

Рис.4 Матрица инцидентности гиперграфа

1. Задача компоновки, классификация, критерии. Используемые модели, в задаче компоновки.

Рассмотрим задачу компоновки. Задача может решаться как **сверху вниз.** Дана схема и заданы типоразмеры платы число посадочных мест на плате и число контактов разъема платы, через которые элементы, попавшие на плату, соединяются электрически с элементами с других плат. Надо разрезать схему на части, соответствующие одной плате.

Задача может решаться снизу вверх. Задана схема и набор готовых плат (микросхем), надо покрыть схему модулями из заданного набора. Первый вариант называют задачей разрезания, а второй – задачей покрытия.

1. **Разрезание** – разбиение схемы на части , критерием качества является минимальное число внешних связей между частями.
2. **Покрытие** –покрытие схемы с использованием заданного набора микросхем с минимизацией числа межмодульных связей и максимальной повторяемостью используемых модулей.

Начнем с выбора модели для задачи разрезания. Есть две модели схемы мультиграф и гиперграф. Оценим их адекватность схеме в задаче разрезания. Предположим, что мы разбиваем фрагмент схемы , представленный на рис.1 на две части, (синяя линия) в первую войдут элементы с номерами 1,2,5 – 1часть, а во вторую часть оставшиеся элементы с номерами 3,4,6. Если оценить результат по мультиграфу , представленному на рис.2, то видно, что в сечение попадают 7 ребер, а если посмотреть по фрагменту схемы, представленному на рис.1, то цепей в сечении 5. Вывод : модель мультиграф для задачи разрезания неадекватно отражает схему, удачней для компоновки гиперграф, так как в нем ребро соответствует цепи.

Рассмотрим существующие алгоритмы компоновки. Все множество алгоритмов можно разбить на группы:

* Точные алгоритмы
* Приближенные

1. Последовательные
2. Смешанные
3. Итерационные

**Точные алгоритмы** сводятся к решению задачи дискретного программирования или задачи метода ветвей и границ. Из-за сложности решения и отсутствия возможности учета в них технологических ограничений эти алгоритмы не нашли применения.

**Приближенные алгоритмы –** нашли наибольшее применение, они дают хорошие результаты, но главное позволяют учитывать технологические требования и обеспечивают необходимые при компоновке критерии качества.

**Итерационные алгоритмы** применяются для улучшения либо результатов компоновки, поученных последовательным алгоритмом, либо начального произвольного разрезания .

Осуществляются путем перестановки ( парной или групповой) вершин графа из различных кусков (элементов схемы из разных плат) с проверкой улучшения заданных при перестановке критериев качества.

**Смешанные алгоритмы** используют чаще всего. Начальная компоновка решается последовательным алгоритмом, а улучшение – итерационным.

Далее мы рассмотрим все эти алгоритмы подробнее.

**Идея последовательных алгоритмов:** сначала выбирается первая вершина (обычно с максимальным числом соединений) и помещается на плату, далее последовательно к ней на плату присоединяются другие вершины из числа нераспределенных ( при условии удовлетворения заданным критериям) , таким образом формируется первый кусок графа (модуль, ячейка т т.д.) Затем формируется аналогично второй кусок и так далее до полного разрезания.

1. Алгоритмы компоновки, классификация. Последовательный алгоритм разрезания мультиграфа.

Рассмотрим существующие алгоритмы компоновки. Все множество алгоритмов можно разбить на группы:

* Точные алгоритмы
* Приближенные

1. Последовательные
2. Смешанные
3. Итерационные

**Точные алгоритмы** сводятся к решению задачи дискретного программирования или задачи метода ветвей и границ. Из-за сложности решения и отсутствия возможности учета в них технологических ограничений эти алгоритмы не нашли применения.

**Приближенные алгоритмы –** нашли наибольшее применение, они дают хорошие результаты, но главное позволяют учитывать технологические требования и обеспечивают необходимые при компоновке критерии качества.

**Итерационные алгоритмы** применяются для улучшения либо результатов компоновки, поученных последовательным алгоритмом, либо начального произвольного разрезания .

Осуществляются путем перестановки ( парной или групповой) вершин графа из различных кусков (элементов схемы из разных плат) с проверкой улучшения заданных при перестановке критериев качества.

**Смешанные алгоритмы** используют чаще всего. Начальная компоновка решается последовательным алгоритмом, а улучшение – итерационным.

Далее мы рассмотрим все эти алгоритмы подробнее.

**Идея последовательных алгоритмов:** сначала выбирается первая вершина (обычно с максимальным числом соединений) и помещается на плату, далее последовательно к ней на плату присоединяются другие вершины из числа нераспределенных ( при условии удовлетворения заданным критериям) , таким образом формируется первый кусок графа (модуль, ячейка т т.д.) Затем формируется аналогично второй кусок и так далее до полного разрезания.

Как ранее говорилось задача компоновки возможна в двух вариантах постановки: разрезание и покрытие.

1. **Разрезание** – разбиение схемы на части , критерием качества является минимальное число внешних связей между частями.
2. **Покрытие** –покрытие схемы с использованием заданного набора микросхем с минимизацией числа межмодульных связей и максимальной повторяемостью используемых модулей.

Как отмечалось в предыдущей лекции , здесь важен выбор модели схемы. Рассмотрим задачу компоновки для двух моделей схемы

### 2.1 Последовательный алгоритм разрезания мультиграфа.

**Постановка задачи.**

Дан мультиграф G(X,U), где U – множество ребер, которые могут быть кратными.

⏐X⏐= N – общее число элементов, вершины графа. Мы преобразуем G в подграфы G1,G2,…Gm, где X1,X2,…Xm – множество подмножеств вершин мультиграфа таких, что :

UiXi=X, таких что ∀ i, j Xi ∩Xj=0

(∀ k <= m) ⏐Xk⏐ <= n, где n –число вершин подграфа Xk.

n\*m=N;

g(Xj)<=P; (число разъемов задано), а g- степень подграфа, при этих ограничениях минимизируется суммарное число соединений между подграфами:

(∀ j, k <= m) Σ g(Xj , Xk) минимизируется, где j не равно k.

**Основные этапы алгоритма.**

1. Выбор начальной вершины.
2. Выбор очередной вершины в формируемый подграф.
3. Проверка выполнения ограничений ⏐Xk⏐ <= n, g(Xj)<=P;
4. Проверка на окончание процесса.

**Обозначим**

Xc - множество свободных вершин графа

Xp  - распределенные по подграфам вершины, которые находятся в j –ом подграфе обозначается Xjp, тогда алгоритм запишется:

1. Xp = 0; Xc=N;
2. Формирование j –го блока. J:= 1; Xj =0;
3. Выбор начального элемента X\*, такого что X\* входит в Xc , а g(X\*) имеет максимальное значение.

Xp = Xp U {X\*}

Xj := Xj U {X\*}

Xc:= Xc\ {X\*}

1. Выбор очередного элемента {X\*}, который входит ву максимальное число цепей Xj или оптимизирует какую- то функцию выбора, например отношение его связей к связям из Xc/
2. Если ⏐Xj  U{X\*}⏐ > n , то к п. 7, иначе к п.6
3. Выполнить:

Xp = Xp U {X\*}

Xj := Xj U {X\*}

Xc:= Xc\ {X\*}, далее п.п. 4,5,6

1. Если g(Xj) > P, то коррекцияч с конца, удаляем из списка назначенных элементы до тех пор , пока не станет g(Xj)<=P; далее п. 8.
2. Проверка ⏐Xc⏐=0, то все элементы распределены, переход к п.9, иначе j:=J+1; и к п. 3; Xj := 0;

1. Конец.

Достоинство алгоритма разрезания - простой, легко учесть требования разработчика схемы. К недостаткам можно отнести зависимость от выбора начального элемента, поэтому часто используют совместно с итерационным.

1. Итерационные алгоритмы компоновки. Пример.

### 2.4 Итерационные алгоритмы

Метод парных и групповых перестановок. В графе, разрезанном на подграфы, производится парная перестановка элементов разных подграфов с проверкой на каждом шаге приращения числа ребер соединяющих куски. Цель итерации – минимизация числа соединительных ребер. Итерационный алгоритм включает следующие процедуры:

1. Из множества подграфов разрезания выделяют два любых G1(X1,U1), G(X2,U2). Для каждой пары вершин Xi1 и Xj2 оценивается разность между числом новых внешних связей (после перестановки) и старых до перестановки - ΔS=Sпосле-Sдо.
2. Те пары Xi1 и Xj2, для которых ΔS <0 заносятся в список кандидатов для обмена.
3. Из списка выбирается та пара , для которых ⏐ΔS⏐ max, если таких несколько , то переставляется любая пара.
4. Переставленные пары отмечаются и больше в перестановке не участвуют.
5. Для вновь выбранных подграфов повторяется процесс с п. 1.
6. Повторяются п.п. 2,3,4 до тех пор , пока в списке замен не будет пар , для которых ΔS <0 (или по времени, или по числу итераций).
7. Далее устраняют все метки вершин в G1 и G2, выбирается другая пара подграфов и повторяются п.п. 1-6, то есть минимизируется число соединений для вновь выбранных подграфов. Процесс продолжается до тех пор , пока не будет сделана перестановка во всех парах подграфов графа G(X,U). Число таких пар равно числу сочетаний m элементов по два m\* (m-1)/2, где m – число подграфов.

Пример. Минимизировать парными перестановками.

До перестановки число межблочных связей равно 5.

Перестановка элементов первого и второго подграфов.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X12 |  | ΔS | X11 |  | ΔS |
|  | X21 | +3 |  | X21 | +2 |
|  | X22 | +4 |  | X22 | +3 |
|  | X23 | +3 |  | X23 | +2 |
|  | X24 | +3 |  | X24 | +2 |
|  | X25 | +3 |  | X25 | +3 |
| X13 |  | ΔS | X14 |  | ΔS |
|  | X21 | +4 |  | X21 | +1 |
|  | X22 | +4 |  | X22 | +3 |
|  | X23 | +3 |  | X23 | +2 |
|  | X24 | +3 |  | X24 | +2 |
|  | X25 | +3 |  | X25 | +3 |

Перестановка элементов первого и третьего подграфов.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X11 |  | ΔS | X12 |  | ΔS |
|  | X31 | 0 |  | X31 | +2 |
|  | X32 | 0 |  | X32 | 0 |
|  | X33 | -1 |  | X33 | +1 |
|  | X34 | -2 |  | X34 | 0 |
| X13 |  | ΔS | X14 |  | ΔS |
|  | X31 | +1 |  | X31 | 0 |
|  | X32 | -1 |  | X32 | -2 |
|  | X33 | 0 |  | X33 | -1 |
|  | X34 | -1 |  | X34 | 0 |

Кандидаты на перестановку X11  и X34 или X32  и X14, так как для них минимальное ΔS

1. Гиперграф как модель схемы в алгоритмах разрезания. Сравнение с моделью мультиграфа.

Начнем с выбора модели для задачи разрезания. Есть две модели схемы мультиграф и гиперграф. Оценим их адекватность схеме в задаче разрезания. Предположим, что мы разбиваем фрагмент схемы , представленный на рис.1 на две части, (синяя линия) в первую войдут элементы с номерами 1,2,5 – 1часть, а во вторую часть оставшиеся элементы с номерами 3,4,6. Если оценить результат по мультиграфу , представленному на рис.2, то видно, что в сечение попадают 7 ребер, а если посмотреть по фрагменту схемы, представленному на рис.1, то цепей в сечении 5. Вывод : модель мультиграф для задачи разрезания неадекватно отражает схему, удачней для компоновки гиперграф, так как в нем ребро соответствует цепи.

I

2

3

4

5

6

1

1

6

5

4

3

2

1

II

III

VII

IX

IV

VIII

VI V

Представим этот фрагмент схемы **мультиграфом** , как на рис.2

1 2

3

4

5

6

Рис.2 Модель фрагмента схемы – мультиграф.

1. Формализация задачи разрезания гиперграфа.

ПОКА НЕТ

**Постановка задачи.**

Дан мультиграф G(X,U), где U – множество ребер, которые могут быть кратными.

⏐X⏐= N – общее число элементов, вершины графа. Мы преобразуем G в подграфы G1,G2,…Gm, где X1,X2,…Xm – множество подмножеств вершин мультиграфа таких, что :

UiXi=X, таких что ∀ i, j Xi ∩Xj=0

(∀ k <= m) ⏐Xk⏐ <= n, где n –число вершин подграфа Xk.

n\*m=N;

g(Xj)<=P; (число разъемов задано), а g- степень подграфа, при этих ограничениях минимизируется суммарное число соединений между подграфами:

(∀ j, k <= m) Σ g(Xj , Xk) минимизируется, где j не равно k.

Исправить с ниже

Алгоритм подобен алгоритму разрезания мультиграфа, разница состоит лишь в выборе очередной вершины. Все ребра делятся на внутренние , соответствующие цепям, которые целиком вошли в кусок и внешние , под них необходим контакт разъема. Правило выбора очередной вершины - максимальное число связей с гиперребрами из куска, либо такая , которая переведет ребро во внутреннее.

**Замечание**

Мы рассмотрели алгоритмы, где все элементы имеют одинаковые габариты, в случае разногабаритных элементов следует переходить к вершинно-взвешенному графу. Вес положительный и имеет смысл площади элемента, соответствующего данной вершине. Тогда задача формулируется так:

Разрезаем G(X,U) при условиях :

(∀ x Є X) h(x)>0

(∀ k <= m) ⏐Xk⏐ <= n

(∀ k <= m) ΣH(xi)<=S (площадь платы), для всех xi Є Xk

При этом минимизируется суммарное число цепей в разрезе. Алгоритм последовательный , контролируется суммарная площадь элементов, число цепей , требующих контактов разъема. В остальном алгоритм похож на алгоритм разрезания мультиграфа или гиперграфа. Основная проблема – это существует возможность отсутствия геометрического размещения элементов.

1. Алгоритм последовательного разрезания на блоки (модель схемы гиперграф).

ПОКА НЕТ

Задача разрезания гиперграфа G(X,U) на m кусков Gi (Xi,Ui), i Є I= {1,2,3 …m}

Совокупность {Gi (Xi,Ui)} – A(G) – разрезание гиперграфа G(X,U), если выполняются следующие условия:

1. (∀i Є I ) [Gi Є A(G) Xi ≠ ∅]
2. (∀i, j Є I ) [Gi Є A(G) & G j  A(G ) 🡪 Xi ∩ Xj≠ ∅]
3. U Xi=X, для всех i Є I
4. (∀i Є I )⏐Xi⏐<=B, где B – целое число.
5. Число цепей , выходящих из куска <= P

Выведем формулу для числа внешних цепей, выходящих из куска. Обозначим:

Xij =1, если i-ый элемент графа Є J-ому куску

0, в противном случае

Сsi = 1, если i-ый элемент Є S-ой цепи

0 – в противном случае

ΣСsi– сумма элементов i – ой цепи, где I =1..N.

ΣCsiXij - число элементов s –ой цепи в j- ом куске, если это число равно сумме элементов цепи, то цепь внутренняя, если нет, то внешняя.

Введем функцию Ψ(z).

Ψ(z) = 1, если z не равняется 0,

0, если z =0

Ψ( ΣСsi - ΣСsi Xi j )= 1, если цепь не в одном куске

0, если цепь в одном куске

Для всех i=1..N

Tj = Σs Ψ( ΣiСsi - ΣiСsi Xi j ) , если i=1..N, а s=1..⏐L⏐ -число цепей внешних для j куска.

Таким образом, условие 5 выглядит так:

Σs Ψ( ΣiСsi - ΣiСsi Xi j ) <=P

Это условие разрезания, но мы минимизируем общее суммарное число межблочных связей .

ΣjΣs Ψ( ΣiСsi - ΣiСsi Xi j ), где j изменяется от1 до m, где m – число кусков разрезания.

Окончательно задача разрезания гиперграфа выглядит так:

Поиск ⏐⏐Хij⏐⏐ при следующих условиях:

Σ jXij =1; j=1..N;

Σ j Xij <=B, где j= 1, m, а i=1..N;

Tj <= P;

Пробуем учесть в формуле случай, когда цепь целиком входит в блок.

Ψ(ΣiСsi Xi j ) - определяет есть ли в j- ом блоке s –ая цепь, 1- есть, 0 – нет.

Σ j (Ψ(ΣiСsi Xi j ) ) – во сколько блоков входит цепь. Если сумма равна 1, то в одну цепь и для нее не нужно контакта разъема. Учесть факт, что цепь внутренняя можно с помощью следующей формулы.

Ψ(Σ j (Ψ(ΣiСsi Xi j ) - 1 ) - во сколько блоков входит цепь.

ΣsΨ(Σ j (Ψ(ΣiСsi Xi j ) - 1 ) функция для минимизации. Здесь s- число цепей s=1..⏐ L⏐

1. Задача компоновки. Алгоритмы покрытия. Эвристический алгоритм задачи покрытия.

Первым этапом технического проектирования печатных плат является задача распределения базовых элементов (БЭ) логической схемы по корпусам микросхем (ИС). Ее можно решать в такой последовательности:

1. Определить оптимальный набор ИС для проектирования схемы из числа разрешенных на данном предприятии.
2. Распределить БЭ по корпусам ИС таким образом, чтобы минимизировать число межмодульных связей.

Мы рассмотрим случай, когда оптимальный набор есть. Это задача покрытия . Ее суть содержательно в следующем.

Исходная схема представляет собой множество связанных функциональных элементов И, ИЛИ,НЕ, триггеров, усилителей, генераторов и т.д. Каждый функциональный элемент реализует некоторую функцию. Их множество определяет совокупность всех функций, выполняемых в схеме.

### 6.1 Эвристический алгоритм задачи покрытия.

Для каждого подграфа Gi(Yi,Ui) – микросхема, в графе G(X,U) ищут изоморфные подграфы, найденные подграфы удаляются из графа G, если после этого остаются непокрытые части Hj, то эта задача решается для каждой части. Процесс продолжается до тех пор, пока вся схема не окажется покрытой модулями из заданного набора. Качество покрытия как всегда в последовательных алгоритмах зависит от порядка выбора элементов из библиотеки модулей. Улучшение покрытия возможно итерационным алгоритмом, при этом переставляются элементы однотипные функционально из различных модулей ,цель – минимизация внешних связей.

Замечания

* Алгоритм может использовать как связные, так и несвязные модули. В последнем случае алгоритм проще.
* Граф может быть ориентированным, если при изоморфизме следует различать направление воздействия типа А на В или наоборот.
* Модель – всегда взвешенный граф, где вес – функция элемента.
* Считается , что все связи модуля выведены наружу корпуса микросхемы, то есть если функциональные элементы микросхемы не соединены-это модули наборы, если соединены-связные модули.

Рассмотрим **эвристический алгоритм покрытия связными модулями**.

1. Из графа G(X,U) выбирается одна из непокрытых вершин.
2. Просматривается библиотека модулей {Gj (Yj,Uj)} среди этих модулей ищем первый, который содержит вершину нужного типа. Если таких несколько, то берем первую и ставим ее в соответствие покрываемой вершине.

G – схема Gj - модуль

5 6

b c

2 3 4

a

d

1

Пусть выбрали вершину 1 ей в модуле выбрали соответствующую вершину **а**, которая по функции совпадает с вершиной 1и по степени тоже, обе име5ют по три связи.

1. Строим единичную окрестность в графе вокруг 1 вершины и в графе Gj вокруг вершины **а.** В окрестность входят все вершины смежные для рассматриваемых вершин и не вошедшие в другие подграфы. Так в графе G - это О1(2,3,4),,a в графе Gj для а – Оа(b,c,d)
2. Сравниваем вершины окрестности покрываемого графа последовательно с вершинами окрестности покрывающего графа и формируем множество вершин Р (это пересечения), для которых совпадают и степени и типы вершин для покрывающего и покрываемого подграфов. В этом случае пересечение окрестностей считается не пусто, но оно может быть и пустым. При сопоставлении вершин окрестностей типы и степени вершин должны совпадать . Например, 2 🡪 c, а 4🡪 b, так как степени и ориентация входящих ребер совпадает и типы тоже. Если все вершины покрывающего подграфа использованы, то соответствующий ему подграф исключается из G(X,U).

Для простоты сравнения элементов покрываемого и покрывающего графов применяют прием ранжирования, то есть при разорванных обратных связях (если они есть),элементы распределяются по ярусам ,есть специальные алгоритмы ранжирования и суть их в следующем.

* В нулевой уровень включают все внешние контакты схемы
* Первый уровень – это элементы , которые получают сигналы с нулевого уровня ит.д.
* I - ый ярус получает сигналы с i-1 яруса

Ранжирование производят при разорванных обратных связях схемы, при построении пересечения графов надо учитывать и сравнивать количество входов и нагрузку на выход элемента.

1. Последовательно - приближенный алгоритм покрытия схемы по шагам.(связные модули).
2. Ранжирование схемы.
3. Выбор начального элемента
4. Выбор покрывающего блока
5. Выбор элемента в покрывающем блоке
6. Построение единичной окрестности вокруг покрывающего и покрываемого элементов.
7. Определение пересечения окрестностей. Если пустое, то к п.7, иначе проверка корректности пересечения с точки зрения функционирования схемы, хватит ли выводов, все ли цепи покрываемой схемы могут быть реализованы. Если пересечение некорректно, то к п. 7, иначе запоминаем его. Запоминаем пересечение, если модуль не исчерпан, то к п.4, если исчерпан то к п.8.
8. Если покрывающий блок использован не полностью , то к п.4, иначе к п.9.
9. Выбор наилучшего пересечения из числа запомненных пересечений .
10. Если схема еще не покрыта, то к п. 2, иначе конец.
11. Покрытие схемы модулями “россыпью”.

Есть множество модулей G1, G2,… Gm. Каждый из этих модулей состоит из несвязанных частей Gi = {g1,g2,…gn}. Пусть ai1- число частей g1 в Gi, то есть в модулях элементы могут повторяться. Задача покрытия сводится к определению вспомогательного набора покрывающих элементов изо всех {Gi}, то есть подбираются gi подходящего типа. Решается задача покрытия схемы блоками из вспомогательного набора, то есть определяется, сколько каких блоков взять. Эта задача решается последовательным алгоритмом. Результатом решения является схема, элементами которой являются блоки вспомогательного набора b1,b2,…bn,.

Каждое значение bl вспомогательного набора, показывает, сколько надо в схеме иметь блоков типа bl . Так как типы блоков в разных модулях библиотеки могут повторяться, то следующей задачей является определение количества библиотечных модулей каждого типа, которые надо иметь, чтобы решить с их помощью задачу покрытия схемы. При этом минимизируется суммарная стоимость.

Пусть Yi –количество блоков, соответствующих подграфам Gi, bi – количество элементов i –го типа вспомогательного набора, которые могут располагаться в разных модулях , соответствующих подграфу Gi. Количество этого типа в Gi обозначим aij. Тогда следующее уравнение покажет сколько элементов первого типа вспомогательного набора b1 будет во всех блоках.

a11y1 +a21y2+ a31y3 + …am1ym >=b1

аналогично для b2

a12y1 +a22y2+ a32y3 + …am2ym >=b2

и так далее для bn

a1ny1 +a2ny2+ a3ny3 + …amnym >=bn

Известно , что каждому Gi  соответствует модуль библиотеки Yi, таких модулей n. Пусть модуль Gi имеет стоимость Сi, тогда надо решить вышеуказанную систему уравнений при условии минимизации функции цели

Так как количество блоков Yi целое число, то мы имеем задачу целочисленного программирования.

После определения сколько каких блоков надо, необходимо определить какие элементы берутся из каких модулей, обязательное требование, чтобы каждый элемент вошел в один из модулей , но так как в модулях содержатся элементы разных типов, то может быть случай что какой-то элемент вспомогательного набора может покрываться не одним исходным модулем. Поступают так:

* Генерируют множество покрытий для каждого элемента
* Решают задачу выбора непротиворечивого покрытия, включающего как можно больше элементов схемы

Для этого строится вспомогательный граф, в котором множество вершин – множество покрывающих кандидатов для всей схемы В графе ребро между вершинами проводится тогда, когда соответствующие элементы вершинам покрытия не могут одновременно войти в результирующее покрытие, но в покрытие должно войти максимальное количество элементов. Это задача нахождения максимального внутренне устойчивого множества в графе. В теории графов существует достаточное количество алгоритмов для решения этой задачи, как приближенные, так и точные. Напомню, что внутренне устойчивое множество графа содержит только не смежные между собой вершины. Рассмотрим алгоритм полного перебора на примере следующего графа:

2 1 3

4

5

6

Рассмотрим следующие варианты несвязанных вершин:

{1,4} {1,5} {2,3,5,6} {2,4} – решением задачи, является максимальное множество {2,3,5,6}

1. Размещение элементов, типы задач размещения. Критерии качества.

### 6.4.Алгоритмы размещения, постановка задачи и критерии

Размещение – это определение местоположения элементов на коммутационном поле конструктивного модуля, такого , при котором создаются наилучшие условия для решения последующей задачи трассировки соединений с учетом конструкторско- технологических требований и ограничений.

Основная сложность в постановке задач размещения заключается в выборе целевой функции. Связано это с тем, что одной из главных целей размещения является создание наилучших условий для дальнейшей трассировки соединений, что невозможно проверить без осуществления самой трассировки. Совместное решение этих задач достаточно сложно.

В настоящее время алгоритмы размещения используют следующие критерии, которые лишь способствуют получению оптимальной трассировки соединений. К таким критериям относятся:

* минимум суммарной длины соединений;
* минимум числа соединений, длина которых больше заданной;
* минимум числа пересечений проводников;
* максимальное число соединений между элементами, находящихся в соседних позициях либо в позициях, указанных разработчиком;
* максимум числа цепей простой конфигурации.
* Число межслойных переходов;
* Параметры паразитных связей между элементами и проводниками
* Равномерность распределения nтемпературы по поверхности подложки.

Наибольшее распространение в алгоритмах размещения получил первый критерий, что объясняется следующими причинами: уменьшение длин соединений улучшает электрические характеристики устройства, упрощает трассировку печатных плат; кроме того, он сравнительно прост в реализации.

Можно выделить следующие типы задачи размещения:

1. Размещение элементов одинаковых габаритов с заранее выделенными однотипными установочными местами. Под однотипными местами понимается, что любой элемент может быть размещен на любое посадочное место.
2. Размещение разногабаритных элементов кратных размеров с заранее определенными установочными местами на коммутационном поле (КП).
3. Размещение разогабаритных элементов, таких как электро-радиоэлементы, компоненты ИС, ГИС и т.д. на КП произвольной конфигурации.

Исходной информацией при решении задач размещения являются: данные о конфигурации и размерах коммутационного пространства, определяемые требованиями установки и крепления данной сборочной единицы в аппаратуре, количество и геометрические размеры конструктивных элементов, подлежащих размещению; схема соединений, а также ряд ограничений на взаимное

расположение отдельных элементов, учитывающих особенности разрабатываемой конструкции. Задача сводится к отысканию для каждого размещаемого элемента таких позиций, при которых оптимизируется выбранный показатель качества и

обеспечивается наиболее благоприятные условия для последующей трассировки. Наиболее часто используется критерий минимальной суммарной длины. Расстояния при этом могут рассчитываться по следующим формулам:

dij=√(xi-xj)2-(yi-yj)2,

что соответствует прокладке соединения по кратчайшему пути.

Dij= + ⏐yi- yj ⏐

-когда соединения прокладываются по каналам и магистралям параллельно осям координат.

Алгоритмы размещения делятся на следующие группы:

* Силовые
* Последовательные
* Итерационные

1. Классификация алгоритмов размещения.

Размещение – это определение местоположения элементов на коммутационном поле конструктивного модуля, такого , при котором создаются наилучшие условия для решения последующей задачи трассировки соединений с учетом конструкторско- технологических требований и ограничений.

Алгоритмы размещения делятся на следующие группы:

* Силовые
* Последовательные
* Итерационные

### 6.5 Силовые алгоритмы размещения

В основу этой группы алгоритмов положен динамический метод. Процесс размещения элементов на плате представляется как движение к состоянию равновесия системы материальных точек (элементов), на каждую из которых действуют силы притяжения и отталкивания, интерпретирующие связи между размещаемыми элементами. Силы притяжения, действующие между любыми двумя материальными точками ri и rj , пропорциональны числу электрических связей между данными конструктивными элементами. Силы отталкивания материальных точек друг от друга и от границ платы исключает возможность слияния двух любых точек и способствует их равномерному распределению по поверхности монтажного поля. Чтобы устранить возникновение в системе незатухающих колебаний, вводят силы сопротивления среды, пропорциональные скорости движения материальных точек.

Таким образом, задача оптимального размещения элементов сводится к нахождению такого местоположения точек, при котором равнодействующие всех сил обращаются в нуль. К достоинствам данного метода относятся возможность получения глобального экстремума целевой функции, а также сведение поиска к вычислительным процедурам, для которых имеются разработанные численные методы.

Недостатками являются трудоемкость метода и сложность его реализации(подбора коэффициентов для силовых связей); необходимость фиксирования местоположения некоторого числа конструктивных элементов на плате для предотвращения большой неравномерности их размещения на отдельных участках платы. Алгоритмы этой группы сложны для реализации на ЭВМ, а также для учета технологических ограничений.

### 6.6 Итерационные алгоритмы размещения

Итерационные алгоритмы имеют структуру, аналогичную итерационным алгоритмам компоновки, рассмотренным ранее. В них для улучшения исходного размещения элементов на плате вводят итерационный процесс перестановки

местами пар элементов. Алгоритмы данной группы характеризуются достаточно высоким быстродействием. Алгоритмы с групповыми перестановками элементов на

практике используются редко ввиду их сложности, которая часто не оправдывает достигаемую степень улучшения результата.

Алгоритмы данной группы предполагают наличие начального размещения, полученного с помощью алгоритмов последовательного размещения или случайным образом. Например, в итерационном алгоритме парных перестановок каждый элемент меняется с каждым другим и оценивается уменьшается ли при этом средняя длина соединений , фиксируется та перестановка, которая дает максимальное уменьшение. Время решения пропорционально n2, поэтому применение метода для больших схем нерационально.

**Модификация:** рассматриваются только соседи, например, при расстоянии к=4,здесь просматривается ~ 2k(k+1) , то есть около 40 перестановок, применяются и перестановки группами.

### 6.7 Последовательные алгоритмы размещения

Последовательные алгоритмы основаны на допущении, что для получения оптимального размещения необходимо в соседних позициях располагать элементы, максимально связанные друг с другом. Сущность этих алгоритмов состоит в последовательном закреплении заданного набора конструктивных элементов на коммутационной плате относительно ранее установленных. В качестве первоначально закрепленных на плате элементов обычно выбирают разъемы, которые искусственно «раздвигают» до краев платы. При этом, все контакты разъемов равномерно распределяются по секциям (столбцам и строкам координатной сетки). На каждом l-ом шаге (l=1,2,…,n) для установки на коммутационную плату выбирают элемент из числа еще не размещенных, имеющий максимальную степень связности с ранее закрепленными элементами При этом определяется пара позиция - элемент, для каждой примерочной позиции элемента определяется суммарная длина соединений элементов, для которой эта позиция наиболее удобна (в смысле минимальной длины соединений с размещенными элементами. Иногда подсчитывается не суммарная длина соединений, а приращение длин соединений то есть вводится функция, учитывающая связи данного элемента с размещенными. После размещения очередного элемента (модуля) процесс повторяется для оставшихся элементов и вакантных установочных мест до тех пор, пока не будут размещены все элементы.

Алгоритмы, использующие последовательный процесс закрепления элементов в позициях, являются в настоящее время самыми быстродействующими. Однако, по качеству получаемого решения последовательные алгоритмы уступают итерационным. Поэтому их используют обычно для получения начального размещения элементов на плате. Результат зависит от начального выбора. В следующей лекции мы на примере рассмотрим по шагам выполнение последовательного размещения с последующим улучшением с помощью итерационного алгоритма.

1. Последовательно-итерационный алгоритм размещения. Последовательный этап. Пример.

### 8.1 Размещение последовательно – итерационным алгоритмом

Дана схема электрическая принципиальная , представленная мультиграфом G(V,R) с числом вершин ⏐V⏐=n. Надо разместить схему в монтажном пространстве Т коммутационной платы с числом посадочных мест m> n/ Критерий минимум суммарной длины соединений:

Q = 1/2Σlij\*Sij,

где lij – расстояние между I и j установочными местами, в которых расположены соответствующие конструктивные элементы, Sij – число кратных связей между элементами на i и j местах. Для оценки расстояния используется формула:

lij=⏐xi- xj ⏐+ ⏐yi- yj ⏐

Алгоритм состоит из двух частей:

* Последовательная
* Итерационная.

На первом этапе последовательно заполняются установочные места элементами с учетом связей между ними, этап заканчивается , когда все элементы размещены.

На втором этапе проводится перестановка элементов с целью минимизации выбранного критерия оптимальности.

1. Зададим множество конструктивных элементов V и связи между ними R графа G(V,R) матрицей смежности S={Sij}nxn,

где n –число элементов на плате

Монтажное пространство задается множеством установочных мест Т в виде матрицы длин L или списка координат позиций. Некоторые элементы могут быть закреплены за отдельными позициями.

1. Размещение производится с помощью коэффициента связности K(Vi), элемента Vi сранее размещенными элементами, определяемого по формуле:

K(Vi) = p(vi ЄVразм) – p(vi ЄVост) =2p (vi ЄVразм) - P(Vi)= 2ΣSij- p(vi)

Здесь ΣSij - число связей элемента Vi с подмножеством размещенных вершин Vразм,

а p(vi ЄVост) – число связей элемента Vi с подмножеством оставшихся вершин.

Jразм  - множество размещенных вершин.

p(vi) – степень вершины vi

Заполнение монтажного пространства:

1. Размещаются закрепленные за установочными местами элементы.
2. В любую свободную позицию размещаются один (первый) конструктивный элемент, далее производится размещение элементов с максимальными значениями коэффициента связности в очередных свободных позициях. В результате такого последовательного решения из элементов формируется массив пар {Vi,Tk}, i=1,n, k=1,m для которого рассчитывается значение критерия Q =

### 8.2. Пример последовательно-итерационного алгоритма размещения.

Дан мультиграф.

3

V1

V4222222222

V83333333333333

V2

V5

V9

V33

V6

V7

3 **1 2 3 4 5 6 7 8 9**

**1**

**2**

**3**

**4**

**5**

**6**

**7**

**8**

**9**

5 0 0 0 3 0 0 2 3 0

2 1 0 0 2 0 2 0 0 0 0

4 0 2 0 1 0 0 0 0 0

3 0 1 0 0 5 0 0 0

2 0 2 0 0 0 2 0 0 4

2 0 0 0 5 2 0 5 0 0

2 2 0 0 0 0 5 0 6 2

2 6 3 0 0 0 0 0 6 0 0

0 0 0 0 4 0 2 0 0

5

На рисунке изображен мультиграф – модель некоторого фрагмента схемы, рядом представление его матрицы , веса указывают число ребер между вершинами.

Монтажное пространство.

|  |  |  |
| --- | --- | --- |
| 7  4  2  1 |  | 3 |
|  | 6 |  |
|  | 5  8 | 9 |

3

2

1

1 2 3

В условии не задано начальное размещение, поэтому выберем элемент V1 и назначим на посадочное место t1, а далее последовательно размещаем все остальные элементы, для этого вычисляем k(v1) коэффициенты связности v1 со всеми остальными по формуле , считая что v1 находится на месте t1.

K1(V2) = 2S21 – P(V2) = 2 \*0 – 4 = -4

Здесь верхний индекс это номер шага. Далее рассчитываются коэффициенты связности первой вершины со всеми остальными.

K1(V3) = 2S31 – P(V3) = 2 \*0 – 3 = -3

K1(V4) = 2S41 – P(V4) = 6-9= -3

K1(V5) = 2S51 – P(V5) = 0 – 8 = -8

K1(V6) = 2S61 – P(V6) = 0 – 12 = - 12

K1(V7) = 2S71 – P(V7) = 4 - 15= -11

K1(V8) = 2S81 – P(V8) = 6 – 9 = - 3

K1(V9) = 2S91 – P(V9) = 0 – 6 = - 6

Максимальный коэффициент связности имеют V3,V4,V8, поэтому их целесообразно разместить рядом в соседние ячейки.

|  |  |  |
| --- | --- | --- |
| V1 | V3 |  |
| V4 | V8 |  |
|  |  |  |

Остались неразмещенными V2,V5,V6,V7,V9 для них аналогично определяют коэффициенты связности с ранее размещенными элементами. Шаг второй:

K2(V2) = 2(S21+S23+S24+S28) – p(V2) = 4 - 4 = 0

K2(V5) = 2(S51+S53+S54+S58) – p(V5) = 0 - 8 = - 8

Представим расчеты в виде таблицы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | K2(Vi) | K3(Vi) | K4(Vi) | K5(Vi) |
| 1 | - | - | - | - |
| 2 | 0 | 0 | 0 | - |
| 3 | - | - | - | - |
| 4 | - | - | - | - |
| 5 | -8 | -8 | -4 | 0 |
| 6 | -2 | 8 |  | - |
| 7 | 1 |  | - | - |
| 8 |  | - | - | - |
| 9 | -6 | -2 | -2 | -2 |

На третьем шаге максимум у V6, на 4-ом максимум у V2, на 5-ом шаге максимум у V5. Отсюда видно, что максимальную связность с V1,V3,V4,V8 имеет элемент V7его размещаем в следующую свободную ячейку монтажного пространства , то есть в номер 3 и т.д. результат получим как на рисунке.

разместить рядом в соседние ячейки.

|  |  |  |
| --- | --- | --- |
| V1 | V3 | V7 |
| V4 | V8 | V6 |
| V2 | V5 | V9 |

1. Последовательно-итерационный алгоритм размещения. Итерационный этап. Пример.

### 8.3 Продолжение примера .Итерационный этап

Суть итерационного алгоритма – это перестановка элементов, но каких? Выберем такой элемент, это будет элемент, который имеет максимальную среднюю длину соединений относительно того места, где он находится. Определим среднюю длину связи элемента Vi, расположенного в позиции tk .

Lvi(tk) = 1/ρ(vi)(ΣjSij lk(i)r(j)) - Это средняя длина связей между элементами i,j, размещенных соответственно в позициях к и позиции r. , тогда для элемента V1, размещенного в позиции t1 средняя длина определяется по формуле:

Lv1(t1) = 1/ρ(v1)(ΣjS1j l1(1)r(j)) = 1/8 (S14l1(1)4(4)+S17l1(1)3(7)+S18l1(1)5(8)) = 1/8(3\*1+2\*2+3\*2) = 1,63

Для элемента V2, размещенного в посадочном месте t7 средняя длина связей :

Lv2(t7) = 1/ρ(v2)(ΣjS2j l7(2)r(j)  =1/4 (S23l7(2)2(3)+S25l7(2)8(5))= 1/4(2\*3+2\*1) = 2

Аналогично определяются связи для остальных элементов:

Lv3(t2) = 2,67 Lv4(t4) = 1,67 Lv5(t8) = 1,25 Lv6(t6) = 1,58

Lv7(t3) = 1,67 Lv8(t5) = 2 Lv9(t9) = 1,33

Из всех полученных значений определяем максимальное Lv3(t2) = 2,67 его и будем переставлять, но куда ? Теперь для V3 определяем предположительную позицию для размещения на основе расчета координаты центра тяжести по формуле:

Xi(Vμ) = 1/ρ(vμ)(ΣjSμj Xr(j)) и Yi(Vμ) = 1/ρ(vμ)(ΣjSμj Yr(j) ) точка размещения вершины μ, где ее средняя длина связей без учета “места ” размещения, то есть считая позицию точками, где Xr(j)  и Yr(j) координаты монтажного пространства, где находится элемент Vi (без размера), тогда для элемента V3

X(V3) = 1/ρ(vμ3 (ΣjS3j Xr(j)  = 1/3(S32X7(2)+ S34X4(4)) = 1/3(2\*1+1\*1) = 1

Y(V3) = 1/ρ(v3)(ΣjS3j Yr(j)  = 1/3(S32Y7(2)+ S34Y4(4)) = 1/3(2\*1+1\*2) = 1,33

Таким образом, для V3 положение центра тяжести Х=1, Y = 1,33, другими словами длина связей будет минимальна при расположении V3 в координатах центра тяжести. Из

позиций расположенных в окрестности центра тяжести формируется окрестность {t4,t7}, в которой размещены элементы V4, V2.  Для определения в какую позицию поместить элемент V3 его примеривают сначала на позицию t4, а затем на позицию t7, то есть пытаются обменять его с V4, затем с V2. Аналогично просчитывают среднюю длину соединений при перемещении V4 на V3 и V2 на место V3.

Lv3(t4(4)) =1/ρ(v3)(ΣjS3j l4(3)r(j) ) = 1/3(S32l4(3)7(2)+S34l4(3)2(4))= 1/3(2\*1+1\*2+3\*2) = 1,33

На место второго Lv3(t7(2)) = 2,33

Lv4(t2(3)) = 1,67 для 4-го элемента при перестановке его с V3

Lv2(t2(3)) = 2,5 для 2-го элемента при перестановке его с V3

Оцениваются изменения средней длины от перестановки Vμ и Vc.

ΔL(Vμ,Vc) = Lv,μ(Tk) - Lvμ(Tc(c)) + Lvc(Ti) – Lvc(Tk(μ)

Вычисляем ΔL(V3,V4) = 1,34 и ΔL(V3,V2) = 0,16 максимальный выигрыш при обмене V3  и V4 при этом суммарная длина 58. Перестановки заканчиваются, когда нет уменьшения длин связей.

1. Алгоритмы трассировки Общая характеристика задачи, зависимость критериев от технологии реализации элементов.

**Алгоритмы трассировки**

Трассировка монтажных соединений – это задача геометрического построения ан КП всех цепей данного конструктива, координаты начала и конца которых определены при размещении элементов. При этом учитываются различные конструктивно – технологические ограничения :допускаются пересечения или нет , возможен ли переход со слоя на слой , сколько слоев разрешено использовать, ортогональная трассировка или нет, допустимая ширина проводников, расстояние между ними и т.д.. Алгоритмы трассировки зависят от принятой конструкции и технологии изготовления РЭА.

1. Трассировка проводных соединений наиболее проста , так как проводники изолированы друг от друга, не надо думать об ограничениях на пересечения , поэтому достаточно минимизировать длину соединений. Но для высокочастотных устройств необходимо учитывать электромагнитную несовместимость цепей.
2. Трассировка печатных и пленочных соединений родственны и зависят от **метрических** ограничений таких как, размеры элементов, ширина и предельные длины проводников и др. Топологических ограничений, например возможность пересечений методы перехода со слоя на слой. Так для МПП самым ответственным является определение координат переходных отверстий, так как возможно их регулярное и нерегулярное расположение.

В зависимости от технологии можно выделить три способа прокладки цепей на коммутационном поле:

* Цепь реализуется в одном слое. Так делают в МПП при технологи с открытыми контактными площадками или выступающими выводами, при однослойной коммутации ИС.
* Цепь реализуется в нескольких слоях, но переходы со слоя на слой организуются через открытые контактные площадки для выводов устанавливаемых элементов.
* Цепь реализуется в нескольких слоях , но переход происходит только через специальные переходные отверстия, число которых может быть значительным и месторасположение их произвольно.

Критерии трассировки зависят от технологии:

|  |  |
| --- | --- |
| 1. Проводной монтаж | Средняя длина соединений, длина самой длинной цепи. |
| 1. Двухсторонние и многослойные ПП со сквозными металлизируемыми отверстиями. | Средняя длина соединений (СДС), число переходных отверстий. |
| 1. МПП с открытыми контактными площадками, МПП с выступающими выводами. | СДС, минимальное число слоев |
| 1. Гибридные интегральные схемы | Число пересечений минимально |
| 1. ИС,СИС, БИС | Число межслойных переходов, число пересечений проводников. |

ОТСЮДА ВРОДЕ НЕ НАДО

Задачу трассировки обычно разделяют на ряд подзадач:

**Расслоение** - распределение соединений по слоям с учетом введения критериев несовместимости цепей с последующей реализацией в каждом слое.

**Трассировка всех цепей на одной плоскости –** при условии минимизации числа пересечений разных цепей с последующим назначением переходных отверстий , расслоение цепей и трассировка цепей на каждом слое.

**Последовательная реализация цепей по слоям**, начиная с коротких, затем длинных и назначение мест переходных отверстий.

**Двухслойная трассировка -** например, в матричных БИС межслойный переход разрешается в любой точке, рассматриваются слои парами, в одном слое цепи проводятся в горизонтальном направлении, а в другом в вертикальном, точка перегиба проводника, является местом переходного отверстия.

**Трассировка по магистралям (каналам) -**  Исходя из пропускной способности канала.

Известные алгоритмы трассировки печатных плат можно условно разбитьна три большие группы:

* + **Волновые** алгоритмы, основанные на идеях Ли и разработанные Ю.Л. Зиманом и Г.Г. Рябовым. Данные алгоритмы получили широкое распространение в существующих САПР, поскольку они позволяют легко учитывать технологическую специфику печатного монтажа со своей совокупностью конструктивных ограничений. Эти алгоритмы всегда гарантируют построение трассы, если путь для нее существует;
  + **Ортогональные** алгоритмы, обладающие большим быстродействием, чем алгоритмы первой группы. Реализация их на ЭВМ требует в 75-100 раз меньше вычислений по сравнению с волновыми алгоритмами. Такие алгоритмы применяют при проектировании печатных плат со сквозными металлизированными отверстиями. Недостатки этой группы алгоритмов связаны с получением большого числа переходов со слоя на слой, отсутствием 100%-ой гарантии проведения трасс, большим числом параллельно идущих проводников;
  + **Алгоритмы эвристического типа**. Эти алгоритмы частично основаны на эвристическом приеме поиска пути в лабиринте. При этом каждое соединение проводится по кратчайшему пути, обходя встречающиеся на пути препятствия.
  + **Топологические**, которые не учитывают метрические ограничения и базируются на алгоритмах анализа планарности графов.
  + Алгоритмы **гибкой трассировки**

**Особняком стоят алгоритмы проводных соединений** Трассировка проводных соединений сводится к построению на фиксированных вершинах контактов элементов, дерева с минимальной суммарной длиной ребер. Используется алгоритм Прима , который реализует следующие принципы:

* Всякая изолированная вершина соединяется с ближайшей вершиной
* Всякая связанная группа вершин соединяется с ближайшей вершиной кратчайшим ребром

Таким образом на контактах цепи строится кратчайшее связывающее дерево, то есть связный граф без циклов.

1. Алгоритм Прима.

**Алгоритм Прима**

Существуют разные алгоритмы соединения контактов – один из них построение кратчайшей связывающей цепи для одной электрической цепи. Известно из теории графов, что кратчайшей связывающей сетью для заданного расположения контактов цепи является дерево. Для построения дерева можно воспользоваться его определением.

**Определение 1**. Дерево это связный граф без циклов.

**Определение 2.** Дерево – это граф, в котором число ребер на единицу меньше чем число вершин.

**Определение 3**. Дерево – граф без циклов, но при добавлении к нему хотя бы одного ребра появляется цикл.

Для построения кратчайшего связывающего дерева воспользуемся следующей процедурой.

1. Для заданного расположения вершин строится полный граф из n вершин и n(n-1)/2 ребер.
2. Упорядочиваем ребра по неубыванию длины и в цикле, начиная с самого короткого, включаем n-1 ребро, получаем КСС.

**Пример**

Есть 6 контактов, координаты их известны в результате решения задачи размещения. Строим на них полный граф как на рисунке. Есть матрица расстояний как на следующем рисунке, чтобы оставаться в поле целых чисел вычисляем расстояния по следующей формуле :

⏐Xi -Xj⏐+⏐Yi – Yj⏐

2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  | 1 |  |  |
| 4 |  |  |  | 3 |
| 5 |  |  |  |  |
|  |  |  |  | 6 |

Расположение контактов.

Матрица удаления контактов друг от друга.

**1 2 3 4 5 6**

**1** 0 1 4 4 5 6

**2** 0 5 5 6 7

**3** 0 6 7 4

**4** 0 1 6

**5** 0 5

**6** 0 матрица симметричная.

Упорядочим ребра по неубыванию длин

* 1. (1-2),(4-5) -длина 1
  2. (1-3),(1-4),(3-6) - длина 4
  3. (1-5),(2-3),(2-4),(5-6) – длина 5
  4. (1-6),(2-5),(3-4),(4-6) – длина 6
  5. (2-6), (3-5) длина 7

n-1 раз повторяем цикл:

Начиная с самого короткого, присоединяем к фрагменту очередное ребро так, чтобы при его включении не появлялись циклы. Полученное дерево и будет кратчайшей связывающей сетью.

**1 шаг** – минимальная длина = 1, проводим ребра 1-2 и 4- 5, суммарная длина сети =2.

**2 шаг** – следующая минимальная длина ребер = 4. Подсоединяем к фрагменту, содержащему вершины 1,2,4,5. Подходят ребра 1-3 и 1-4, оба длиной =4, теперь суммарная длина дерева =2+4+4 =10.

**3 шаг**. К фрагменту 1-3 подсоединяем 3-6, его длина =4, теперь суммарная длина дерева = 14. Все вершины соединены.

**Компьютерный алгоритм Прима**

Строим матрицу расстояний.

**1 2 3 4 5 6**

**1** 0 1 4 4 5 6

**2** 0 5 5 6 7

**3** 0 6 7 4

**4** 0 1 6

**5** 0 5

**6** 0

* Выбираем в матрице ребро минимальной длины – это 1-2, если несколько ребер одинаковой длины, то выбираем первое по порядку.
* Заносим это ребро в список проведенных ребер.
* Далее исключаем ребро из матрицы, заменяя его в матрице агрегированной вершиной 1-2.

Пересчитываем матрицу. В нее заносим расстояния до ближайшей из агрегированных вершин, например 3-я вершина с первой – длина 4, а со

* второй длина 5, заносим минимальное значение для агрегированной вершины, получим матрицу:

**1 - 2 3 4 5 6**

**1-2** 0 4 4 5 6

**3** 0 6 7 4

**4** 0 1 6

**5** 0 5

**6** 0

* Ищем в новой матрице ребро с минимальным весом – это 4-5, создаем из него агрегированную вершину и пересчитываем матрицу.

**1 - 2 3 4 - 5 6**

**1-2** 0 4 4 6

**3 4** 0 6 4

**4-5 4 6** 0 5

**6 6 4 5** 0

**Пересчет**: выбираем ребро первое по порядку длиной 4 , сначала смотрим, можем ли соединить фрагменты 1-2,4-5 или подсоединить вершину 3 . После соединения фрагментов пересчет.

**2-1-4-5 3 6**

**2-1-4-5** 0 4 5

**3**  0 4

**6** 0

Процесс продолжается до тех пор, пока матрица не выродится в **0**. Следующий этап добавляем к фрагменту 2-1-4-5 вершину 3, после пересчета имеем

**2-1-4-5 2-1-3 6**

**2-1-4-5** 0 4 5

**2-1- 3**  0

**6** 0

1. Формализация задачи трассировки. Коммутационное поле, соседство, приоритет координат. Классификация алгоритмов.

ПОКА НЕТ

Задачу трассировки обычно разделяют на ряд подзадач:

**Расслоение** - распределение соединений по слоям с учетом введения критериев несовместимости цепей с последующей реализацией в каждом слое.

**Трассировка всех цепей на одной плоскости –** при условии минимизации числа пересечений разных цепей с последующим назначением переходных отверстий , расслоение цепей и трассировка цепей на каждом слое.

**Последовательная реализация цепей по слоям**, начиная с коротких, затем длинных и назначение мест переходных отверстий.

**Двухслойная трассировка -** например, в матричных БИС межслойный переход разрешается в любой точке, рассматриваются слои парами, в одном слое цепи проводятся в горизонтальном направлении, а в другом в вертикальном, точка перегиба проводника, является местом переходного отверстия.

**Трассировка по магистралям (каналам) -**  Исходя из пропускной способности канала.

Классификация алгоритмов :

Известные алгоритмы трассировки печатных плат можно условно разбитьна три большие группы:

* + **Волновые** алгоритмы, основанные на идеях Ли и разработанные Ю.Л. Зиманом и Г.Г. Рябовым. Данные алгоритмы получили широкое распространение в существующих САПР, поскольку они позволяют легко учитывать технологическую специфику печатного монтажа со своей совокупностью конструктивных ограничений. Эти алгоритмы всегда гарантируют построение трассы, если путь для нее существует;
  + **Ортогональные** алгоритмы, обладающие большим быстродействием, чем алгоритмы первой группы. Реализация их на ЭВМ требует в 75-100 раз меньше вычислений по сравнению с волновыми алгоритмами. Такие алгоритмы применяют при проектировании печатных плат со сквозными металлизированными отверстиями. Недостатки этой группы алгоритмов связаны с получением большого числа переходов со слоя на слой, отсутствием 100%-ой гарантии проведения трасс, большим числом параллельно идущих проводников;
  + **Алгоритмы эвристического типа**. Эти алгоритмы частично основаны на эвристическом приеме поиска пути в лабиринте. При этом каждое соединение проводится по кратчайшему пути, обходя встречающиеся на пути препятствия.
  + **Топологические**, которые не учитывают метрические ограничения и базируются на алгоритмах анализа планарности графов.
  + Алгоритмы **гибкой трассировки**

**Особняком стоят алгоритмы проводных соединений** Трассировка проводных соединений сводится к построению на фиксированных вершинах контактов элементов, дерева с минимальной суммарной длиной ребер. Используется алгоритм Прима , который реализует следующие принципы:

* Всякая изолированная вершина соединяется с ближайшей вершиной
* Всякая связанная группа вершин соединяется с ближайшей вершиной кратчайшим ребром

Таким образом на контактах цепи строится кратчайшее связывающее дерево, то есть связный граф без циклов.

1. Волновой алгоритм, основные шаги. Пример. Достоинства и недостатки волнового алгоритма.

### 11.1 Волновой алгоритм

Все ячейки монтажного поля подразделяют на занятые и свободные.

Занятыми считаются ячейки, в которых уже расположены проводники,

построенные на предыдущих шагах, или находятся монтажные выводы элементов,

а также ячейки, соответствующие границе платы и запрещенным , для

прокладывания проводников участкам. Каждый раз при проведении новой трассы

можно использовать лишь свободные ячейки, число которых по мере проведения

трасс сокращается.

На множестве свободных ячеек коммутационного поля моделируют волну, которая распространяется из одной ячейки, называемой истоком в другую, называемую стоком, соединяемых впоследствии общим проводником. Первую ячейку, в которой зарождается волна влияний, называют источником, а вторую – преемником волны. Чтобы иметь возможность следить за прохождением фронта волны , его фрагментам на каждом этапе присваивают некоторые веса по какому-то правилу, которое характеризует критерий качества, например , Pk= Pk-1 + 1

где Pk и Pk-1 - веса ячеек k-го и (k-1)-го фронтов; зта формула - весовая

функция, являющаяся показателем качества проведения пути, каждый параметр

которой , характеризует путь с точки зрения одного из критериев качества (длины пути, числа пересечений и т.п.). На Pk накладывают одно ограничение – веса ячеек предыдущих фронтов не должны быть больше весов ячеек последующих фронтов. Фронт распространяется только на соседние ячейки, которые имеют с ячейками предыдущего фронта либо общую сторону, либо хотя бы одну общую точку. Процесс распространения волны продолжается до тех пор, пока её расширяющийся фронт , не достигнет приемника или на каком-то шаге не найдется ни одной свободной ячейки, которая могла бы быть включена в очередной фронт, что соответствует случаю невозможности проведения трассы при заданных ограничениях. При построении фронта в новый фронт включают те ячейки, для которых имеется наилучшее значение критерия качества.

Если в результате распространения волна достигла приемника, то осуществляют «проведение пути», которое заключается в движении от приемника к источнику по пройденным на этапе распространения волны ячейкам, следя за тем, чтобы значения Pk монотонно убывали. В результате получают путь, соединяющий эти две точки. Из описания алгоритма следует, что все условия,

необходимые для проведения пути, закладываются в правила приписывания веса

ячейкам.

Чтобы исключить неопределенность при проведении пути для случая, когда несколько ячеек имеют одинаковый вес, вводят понятие путевых координат, задающих предпочтительность проведения трассы. Каждое направление кодируют двоичным числом по mod q, где q – число , просматриваемых соседних ячеек. При этом чем более предпочтительно то или иное направление, тем меньший числовой код оно имеет. Например, если задаться приоритетным порядком проведения пути сверху, справа, снизу и слева, то коды соответствующих путевых координат будут 00, 01, 10, и 11.

Приписывание путевых координат производят на этапе распространения волны. При проведении пути движение от ячейки к ячейке осуществляют по путевым координатам.

Существенными недостатками волнового алгоритма являются малое быстродействие и большой объем оперативной памяти ЭВМ, необходимый для хранения информации о текущем состоянии всех ячеек коммутационного поля,

возможность построения лишь соединений типа «вывод-вывод». Попытки устранить указанные недостатки привели к созданию ряда модификаций волнового алгоритма.

**Анализ задачи.**

Надо хранить, а главнее распознавать L состояний клетки, где L – длина пути плюс хранить занята или свободна клетка. Таким образом, следует распознавать n= 2L+2 комбинаций. Это требует памяти n= log2(L+2); Попытка сократить число распознаваемых и хранимых комбинаций – приоритет координат , который указывает откуда пришла волна, тогда следует хранить только 4 путевых координаты и занята или свободна клетка n= log2 6

**Пример.** Надо построить путь минимальной длины**.** Обозначения

Свободная ячейка

Преграда

Соединяемые точки

Для сокращения числа поворотов, введем следующее правило при переходе от одной ячейки фронта к другой –сохранять предыдущее направление движения, это правило можно задать приоритетом путевых координат.

Pk = Pk-1 если направление движения не меняется и Pk = Pk-1+1, если направление движения изменилось.

1 2 3 4 5 6 7 8 9 10

3 2 1 А 1

4 3 2 1 2

5 4 3

B

6 5 4

7 6 5 9 10

8 7 6 7 8 9 10

9

10

1

2

3

4 путь

5

6

7

8

8

9

1. Основные составляющие волнового алгоритма, позволяющие учитывать заданные ограничения разработчика схемы. Соседство, путевые координаты, весовые функции. Примеры. Способы улучшения волнового алгоритма.

**СЮДА ЕЩЁ ДОБАВИТЬ**

**Сложные критерии трассировки.**

Практически в любой задаче трассировки печатных плат приходится учитывать не один, а несколько критериев качества . Если удается задать количественную оценку важности отдельных критериев , то задача может быть решена с помощью алгоритма Ли. Рассмотрим один из возможных вариантов присвоения веса незанятой ячейки к –го фронта.

1. Рк  = Рк -1 + 1, если в данной и соседних ячейках нет ранее построенных проводников и путевая координата не меняет своего направления.( минимум поворотов).
2. Рк  = Рк -1 + 2, если в соседних ячейках нет ранее построенных проводников , н путевая координата меняет свое направление.
3. Рк  = Рк -1 + 3,если в данной ячейке путевая координата не меняет своего направления и нет ранее построенного проводника , но в соседней ячейке такой проводник есть. (отталкивание от проведенных трасс).
4. Рк  = Рк -1 + 5, если в данной ячейке происходит пересечение с ранее построенным проводником. (минимум пересечений).

Построение пути между точками А и В выглядит как на рисунке. Путь имеет длину 11 ячеек, два перегиба и три пересечения с ранее проведенными проводниками. Возможен другой путь той же длины с тремя перегибами и одним пересечением, но равномерность распределения проводников хуже.

1 2 3 4 5 6 7 8 9 1 0 11

1

2 А

3

4

5

6

7

8 В

9

Здесь на рисунке штриховой линией изображены ранее проведенные проводники, толстая линия – проведенный из точки А в точку В проводник.

**Недостатки параллельной оптимизации**.

* Коэффициенты важности отдельных критериев не всегда могут быть оценены.
* Из-за усложнения расчета весов ячеек дискретного поля время реализации алгоритма существенно возрастает.

Поэтому иногда множество параметров оптимизации упорядочивают в соответствии с важностью каждого критерия и на множестве оптимальных по первому критерию ищут множество оптимальности п второму критерию и так далее. Такую оптимизацию называют условной. Общий алгоритм проведения пути с помощью алгоритма ЛИ выглядит следующим образом:

1. Предварительная обработка исходной информации. На основе конструкторско-технологических ограничений определяют основные критерии , по которым должна вестись оптимизация . Выбирается вариант трассировки. Размечается поле КП выводы элементов, границы рабочего поля. Координаты рабочего поля задают положение ножек микросхемы. Для этого выводы модулей нумеруют и каждой I –ой ножке j- го мдуля ставят в соответствие приращение координат Δх и ΔYотносительно центра его установочной позиции Xi,Yj. Тогда координаты i –ой позиции j-го модуля определяются как Xj+ Δхi, Yj. +ΔYi.
2. Устанавливают очередность трассировки соединений одной цепи. Для этого все пары контактов подлежащих соединению располагают в порядке увеличения расстояний между соседними контактами.
3. Распространяют волну в соответствии с выбранными критериями оптимизации.
4. Строят путь, проходя от стока к источнику по минимальным весам или путевым координатам.
5. Формируют выходные файлы для технологического этапа.

**Способы улучшения волнового алгоритма.**

Существенными недостатками волнового алгоритма являются малое быстродействие и большой объем оперативной памяти ЭВМ, необходимый для хранения информации о текущем состоянии всех ячеек коммутационного поля, возможность построения лишь соединений типа «вывод-вывод». Попытки устранить указанные недостатки привели к созданию ряда модификаций волнового алгоритма.

1. **Ограничение на площадь распространения волны**. В результате уменьшается алгоритмический процент трассировки.
2. **Метод встречной волны**, уменьшается время распространения волны, сокращается машинное время примерно в 2 раза. Источником волны являются обе ячейки, подлежащие электрическому объединению. На каждом шаге поочередно строятся соответствующие фронты первой и второй волны, распространяющихся из этих ячеек. Процесс продолжается до тех пор, пока какая либо ячейка из фронта первой волны не попадет во фронт второй волны. Проведение пути осуществляется из ячейки пересечения фронтов в направлении обеих источников по правилам, описанным в алгоритме Ли. Недостаток этого алгоритма – выделение дополнительной памяти на каждую ячейку рабочей волны для хранения признака принадлежности ее к первой или второй волне, однако недостаток скрашивается выигрышем во времени.
3. **Распространение волны от фрагмента к фрагменту**. Идея состоит в следующем : сначала на контактах одной цепи строится по алгоритму Прима кратчайшее связывающее дерево. Это делается для всех цепей. Затем двух контактные отрезки упорядочиваются , например по длине и в этом порядке трассируются . В этом случае суммарная длина ребер получается меньше обычной. Здесь используется алгоритм Штейнера, построение кратчайшего дерева за счет введения дополнительных точек.

Достоинства волнового алгоритма:

* Универсальный характер алгоритма.
* Использование различных критериев и их сочетаний
* Всегда строит оптимальную трассу, если она существует.

1. Построение путей волновым алгоритмом по заданным параметрам(мин изгибов, мин пересечений и т.д.)

ПОКА НЕТ

1. Топологическая и гибкая трассировка достоинства и недостатки. Модель схемы.

# Топологическая трассировка

Трассировка считается топологической, если:

* не учитываются метрические характеристики. Не рассматриваются ширина проводников и расстояние между ними, размеры платы, а оперируют такими понятиями как лежать между, обход против часовой стрелки, слои и пересечения проводников.
* Методы базируются на плоской укладке проводников, элементы представляются множеством контактов, критериями служат минимальное число слоев и пресечений.
* Для однослойных печатных плат размещение проводников совмещается с задачей размещения элементов.

В 2002 году компания Altium( бывшая Protel) заявила о создании топологического трассировщика.

сложности топологических задач можно судить по следующему примеру. Пусть есть 2 элемента и у них есть три соединения. Для них можно рассмотреть три топологических варианта укладки проводников.

2 3

1 1

3 2

2 3

2 3

1 1 1

1

3 2

3

2

Имеющиеся фотопостроители не умеют работать с топологическими описаниями схемы. Отсюда следует, что необходима метризация, которая иногда не решается без нарушения технологических норм. В результате пришли к выводу, что модель рабочего поля все же должна содержать информацию о метрике такую как допустимую загруженность

трассами промежутков между контактами, а так же позволять определять длины трас и выбирать оптимальную из них, такие методы стали называть топографическими.

Базилевич предложил метод гибкой трассировки, где рабочее поле разбивается на выпуклые многоугольные области – треугольники, в узлах которых находятся контакты. Это модель поля для трассировки, трассировка на таком поле – гибкая трассировка.

трасса

# Достоинства и недостатки топологического метода.

Известно, что традиционные методы не могут давать 100 трассировку по какому-либо критерию качества , например суммарной длине трасс или по числу межслойных переходов. Все существующие САПР с точки зрения трассировки практически не отличаются друг от друга. Такая схожесть систем является следствием схожести применяемых ими алгоритмов, а особенно схожести модели коммутационного поля печатной платы.

В чем плюсы топологической трассировки:

* Для представления данных требуется мало машинной памяти, так как отсутствует дискретное поле.
* Пространство допустимых решений не велико, что можно говорить о поиске точного решения за приемлемое время
* Отсутствует ортогональность разводки, проводники идут как им ближе
* Форма проводников после прокладки не фиксируется, а фиксируется относительное расположение проводников.
* Преодолевается сеточность , так как каждый элемент топологии в старой сетке представлялся набором прямоугольников дискретов, связанных с шагом сетки(расстояние между проводниками), если шаг уменьшается то число дискретов резко возрастает, а следовательно возрастает время и требования к памяти. Если какая- то деталь топологического элемента занимает часть дискрета, все равно занятым считается весь дискрет, трасса тоже занимает весь дискрет, особенно сложно ширина трасс разная или шаг выводов разный.

Теперь о минусах:

* Имеющиеся на производстве фотопостроители не умеют работать без метрики, переход от топологии к метрике называют метризацией, эта не тривиальная задача.

Обычно, печатная плата имеет заранее заданную формау с фиксированным расположением разъемов, технологических отверстий, зон запрета трассировки.

* Поэтому может оказаться так, что найденное топологическое решение не имеет допустимого отображения на заданный конструктив, то есть не может быть реализовано без нарушения технологических ограничений.
* Современные технологии, обычно позволяют проводить трассы дорожки между контактами элементов РЭА или под планарными контактами в другом слое. Если топологический трассировщик не будет это учитывать, то решение будет неоптимальным, то есть топологическая модель неадекватна свойствам реального конструктива.

Перечисленные недостатки привели к тому, что чисто топологические методы применения не получили. Пришли к выводу, что модель рабочего поля все же должна содержать информацию метрического характера: допустимую загруженность трассами промежутков между контактными площадками, возможность определять длины трасс и выделять оптимальные.

### Модель схемы при топологическом анализе .

Модель должна учитывать параметры элементов и коммутационного поля, обусловливающие ограничения на построение соединений.

* Порядок расположения выводов элементов.
* Возможность прохода соединений между ними.
* Расположение внешних выводов на коммутационном поле.

Сформулируем требования к моделям элементов:

1. Граф элемента должен быть планарным
2. Граф элемента должен сохранять порядок следования выводов элемента.
3. В любом плоском предсавлении схемы, чертеж элемента должен быть расположен в области не содержащей моделей других элементов.
4. Модель элемента должна позволять описывать возможности подхода соединений к выводам элемента.
5. Модель элемента должна учитывать возможности прохода соединений между выводами.

Обычно в качестве модели элементов при анализе планарности графа выбирают “звезду” при неупорядоченном наборе выводов или “колесо” ,когда порядок расположения выводов известен возможна и частичная упорядоченность выводов и возможность прохода между ними. Любая цепь представляется графом типа звезда.

Граф типа звезда:

1

2

5

3

4

Граф типа колесо;

1

2

6

3

5 4

1. Многослойная трассировка. МПП с открытыми контактными площадками. Основные задачи автоматизации.

**Трассировка многослойных соединений.**

Высокая плотность компоновки элементов современной РЭА привела к необходимости использования для их коммутации многослойные печатные платы. (МПП) В зависимости от способа выполнения межслойных соединений различают МПП:

* **С открытыми контактными площадками**;
* **Со сквозными металлизированными отверстиями**;

**В первом случае** переход из слоя в слой разрешается только по ножкам модулей. Однако, необходимость построения каждого соединения в пределах одного слоя приводит к увеличению средней длины печатного проводника и к усложнению их конфигурации, в результате чего коэффициент заполнения проводниками каждого слоя получается сравнительно низким, а число слоев платы - относительно большим, поэтому основным критерием оптимизации данного вида МПП является минимум числа слоев платы.

**Во втором случае** разрешены разные участки соединения проводить в разных слоях, а переход со слоя на слой осуществляется через металлизированные отверстия . В чем разница? В первом случае переход в одном заданном месте через открытые площадки, во втором отверстия разрешается иметь в любом месте, что позволяет уменьшить число слоев металлизации по сравнению с предыдущим случаем. Однако, минус этого последнего метода в более низкой надежности , так как требует дополнительной механической обработки и гальванопокрытия контактных переходов. Надежность подобных переходов ниже надежности печатных проводников, что приводит к снижению процента выхода годных плат, а следовательно и росту их стоимости. Поэтому основным критерием оптимизации таких МПП является минимум числа переходов из слоя в слой.

**МПП с открытыми контактными площадками.**

Основной критерий качества – минимум числа слоев. Процесс проектирования разбиваются на три этапа:

1. Построение оптимальных связывающих деревьев для каждой цепи.
2. Разбиение ребер максимального леса на непересекающиеся подмножества.
3. Определение очередности построения соединений каждого слоя платы.
4. Трассировка печатных проводников.

Первый этап – это построение деревьев Прима. Четвертый этап – например, алгоритм Ли. Эти алгоритмы были ранее.

Рассмотрим процесс разбиения ребер на непересекающиеся подмножества. Выделяют горизонтальные и вертикальные ребра. Считают, что ребро ориентировано горизонтально, если выполняются условия:

⏐Xi - Xj⏐>=⏐Yi - Yj⏐

, где Xi, X и Yi,  Yj – координаты вершин,

Если условие не выполняется, то относим это ребро к подмножеству вертикально ориентированных вершин. Такое разбиение ребер на горизонтально и вертикально ориентированные подмножества исключает возможность их пересечение в каждом из отдельно взятых слоев.

Два ребра m1  и m2  не пересекаются, если координаты вершин удовлетворяют одному из условий : пусть для m1 вершины концов ребра (Х I,Yj) и ( Хj,Yj), а для m2 вершины концов (X t,Yt) и (XS, YS), тогда должны быть выполнены:

Xi < Xs, Xi < Xt, Xj < Xs, X j < Xt (1)

или

Xi > Xs, Xi > Xt, Xj > Xs, X j > Xt (2)

Y I < Ys , Yi <Yt, Yj < Ys, Yj < Yt, (3)

или

Y I > Ys , Yi > Yt, Yj > Ys, Yj > Yt,(4)

Предварительное распределение по слоям и определение минимального числа слоев происходит с использованием аппарата теории графов, предварительно поставив в соответствие каждому проводнику вершину некоторого графа. Две вершины графа считаются смежными, если соответствующие им проводники пересекаются. Тогда задача оптимального распределения проводников по слоям МПП сводится к хроматической раскраске вершин графа. Как известно из теории графов это означает ,что никакие смежные вершины графа не могут быть покрашены одинаковым цветом. Количество цветов будет равно числу слоев непересекающихся ребер.

После определения множеств непересекающихся отрезков проводят приблизительную оценку ожидаемого числа слоев. Например, с помощью теории графов. Строят вспомогательный граф, где каждому проводнику ставят в соответствие вершину. Две вершины соединяют ребром, если проводники пересекаются. Решают задачу хроматической раскраски графа , то есть надо раскрасить граф минимальным числом цветов, так что никакие смежные вершины не должны быть окрашены одинаковым цветом. Все вершины одной окраски указывают на проводники, которые должны находиться в одном слое. Такая задача называется поиском хроматического числа k. Для полного графа

k(g)= ⏐x⏐=n

Для любого связного графа без петель и смежных ребер выполняется соотношение:

K(g)<= ρmax(x)+1,

где ρmax(x) – max ρ(xi) для всех вершин xi.

1. Алгоритм анализа планарности. Основные шаги

### Графотеоретический подход к автоматизации конструирования схем

Cодержит следующие этапы:

1. Получение представления схемы в виде графа с учетом конструктивных и технологических ограничений.
2. Анализ планарности графа
3. Получение максимального плоского графа.
4. Планаризация графа схемы.
5. Построение чертежа плоского графа – схемы.
6. Отображение плоского графа в топографию
7. Трассировка, например волновым алгоритмом.

В процессе анализа планарности схемы и выделения плоского подграфа должны быть определены соединения, которые при заданных конструктивно - технологических ограничениях приводят к пересечениям в схеме.

Этап планаризации предусматривает устранение пересечений при использовании дополнительных возможностей: введение перемычек, установка навесных элементов, прохода между выводами элементов и т.д. Как правило, это производится конструктором по результатам анализа планарности схемы в интерактивном режиме.

Отображение в топографию означает переход в метрику.

На начальном этапе, когда происходит анализ планарности графа недостаточно установить планарен граф или нет, так как как правило, графы схем непланарны. Необходимо синтезировать изображение максимального плоского подграфа графа схемы и удалить из него минимальное количество ребер, дающих пересечения. Алгоритм должен учитывать два факта:

* Вершины графа , соответствующие внешним полюсам схемы должны лежать на внешней грани графа.
* Должен соблюдаться заданный циклический порядок следования вершин графа, соответствующих контактам схемы.

Алгоритм состоит из следующих шагов:

1. Построение произвольного цикла в графеG и переноса его в плоское изображение R.
2. Определение пересечения множества вершин, образующих цикл с множеством вершин остатка графаG.
3. Построение цепи между парой соседних вершин плоского графа, которые являются общими для графов R и G.

Проектирование любой схемы предполагает, что заданы внешние полюса схемы, которые целесообразно располагать на границе подложки.

Выведение полюсов на границу соответствует в модели схемы выведению вершин – полюсов графа во внешнюю грань. Задача выделения максимального плоского подграфа является более простой в том случае, если задан порядок следования полюсов.

Соединяя полюса в заданном порядке, получаем цикл, который естественно считать внешней гранью выделяемого плоского графа.

Построение максимального плоского подграфа по заданной внешней грани осуществляется однозначно, если все ребра, принадлежащие максимальному плоскому графу, укладываются внутри цикла

Одним из основных шагов алгоритма построения плоской укладки является поиск цепи между парой вершин (листьев) и укладка ее в грань , которой принадлежат точки начала и конца цепи. Вершины – листья – это вершины общие для графаG(схема) и R – плоское изображение схемы в процессе построения плоской укладки.

Граней для укладки может быть несколько. Выбор грани для укладки цепи на промежуточном этапе существенно влияет на число пересечений в изображении графа на плоскости, поэтому вводится оценка возможного числа пересечений ребер при укладке в ту или иную грань. ВСТАВИТЬ ФОРМУЛУ

Алгоритм включает несколько шагов, на каждом из которых происходит объединение плоского подграфа, полученного на предыдущем шаге с цепью из G конечные вершины которой принадлежат плоскому подграфу R и являются для него открытыми (то есть степень вершины в графе G и в графе R не равны). Концевые вершины вводимой цепи принадлежат графу G и графу R одновременно и в алгоритме называются листьями.

Для поиска цепи в графе G между парой заданных вершин, в графе G из любой вершины “листа ” как из корня , строится вспомогательное дерево Т такое, что листья графа G являются листьями дерева Т, кроме одного листа, на котором как на корне строится это дерево. В дереве Т любая цепь между корнем и листом являются либо парой листьев строится однозначно, причем из способа построения дерева следует, что ни одна из таких цепей не содержит иных листьев , кроме концов цепей. Все цепи между корнем и листьями дерева, а также между каждой парой листьев образуют множество цепей кандидатов для введения в подграф R.

Каждая цепь укладывается в определенную грань, при этом в случае неоднозначности выбирается грань с минимальным значением оценки появления пересечений на последующих шагах укладки.

Оценка работы алгоритма проводилась на основе его реализации на полных и неполных графах. Для полных графов число ребер, удаляемых программой, сравнивается с числом планарности полного графа nп, которое определяется по формуле [1]:

nп=(n-3)(n-4)/2;

здесь n- число вершин графа;

Так как полное дерево является планарным графом, то верхней оценкой числа планарности графа является цикломатическое число K(G), которое определяется для графа G с числом вершин n и ребер m и числом компонент связности p [2 ] :

K(G)= m-n+p;

В табл. 1 представлена зависимость, полученная экспериментально, числа удаляемых ребер графа G при выделении максимальной его плоской части для непланарных полных графов, в зависимости от n и m исходного графа и оценка этого числа в сравнении с числом планарности графа nп.

Таблица 1

Сравнение числа удаляемых алгоритмом непланарных ребер

с оценками (полные графы).

|  |  |  |  |
| --- | --- | --- | --- |
| n-число вершин графа | Число удаленных ребер(эксперимент) | Число планарности nп=(n-3)(n-4)/2 | Число удаленных ребер по K(G) |
| 5 | 1 | 1 | 5 |
| 8 | 10 | 10 | 21 |
| 10 | 21 | 21 | 26 |
| 12 | 36 | 36 | 55 |
| 15 | 66 | 66 | 91 |
| 18 | 105 | 105 | 135 |
| 20 | 136 | 136 | 171 |
| 24 | 210 | 210 | 263 |
| 28 | 260 | 260 | 351 |
| 30 | 351 | 351 | 406 |
| 35 | 496 | 496 | 561 |

В табл.2 представлена, полученная экспериментально для непланарных графов зависимость числа удаленных ребер от n и m и оценки этого числа, полученные по формуле цикломатического числа K(G).

Таблица 2.

Сравнение числа удаляемых алгоритмом непланарных ребер с оценками (неполные графы).

|  |  |  |  |
| --- | --- | --- | --- |
| Число вершин n | Число ребер m | Число удаленных ребер d | D по цикломатическому числу |
| 8 | 23 | 6 | 16 |
| 10 | 40 | 15 | 31 |
| 12 | 45 | 17 | 34 |
| 15 | 62 | 25 | 48 |
| 18 | 83 | 38 | 66 |
| 20 | 91 | 41 | 72 |
| 24 | 157 | 62 | 134 |
| 28 | 221 | 91 | 194 |
| 30 | 248 | 104 | 219 |
| 35 | 296 | 127 | 262 |

Из анализа результатов следует, что для полных графов число непланарных ребер, удаляемых программой совпадает с числом планарности , опредляемого по формуле. Для неполных графов число удаляемых непланарных ребер составляет от 30 до 50 процентов от числа ребер, которые должны быть удалены в соответствии с верхней оценкой по цикломатическому числу.