計算機アーキテクチャ 110−15
設 計 自 動 化 73−15
（1995. 1. 20）

スケッチレイアウトシステムにおける配線可能性検証

田中 秀彦　佐藤 政生　大附 辰夫

早稲田大学理工学部

現在のプリント基板やMCM(Multi Chip Module)のレイアウト設計においては，回路の電気的特性等性能を考慮できる柔軟な設計が要求されている．このような要求に対し，我々はスケッチと呼ばれるトポロジカルなレイアウト表現を用いた柔軟なレイアウトシステムを提案した．このレイアウトシステムの特徴の１つとして，高速な配線可能性検証が挙げられる．本稿では，この配線可能性検証に関する理論およびアルゴリズムを提案し，計算機実験により提案手法の有効性を示す．提案する手法は，従来手法がレイアウト全体に対して行われたのに対し，レイアウト全体をいくつかの領域に分割することによって高速化を目指すものである．

Routability Checking in Sketch Layout System

Hidehiko TANAKA,　Masao SATO, and Tatsuo OHTSUKI

School of Science and Engineering, Waseda University
3-4-1 Ohkubo, Shinjuku, Tokyo 169

Abstract

In printed circuit board and MCM(Multi Chip Module) layout design, flexible design sytems with taking performance such as electric characteristics of wires into consideration are desired. For such request, we have presented a flexible and practical layout system based on a topological layout representation model (sketch model). One of the features of our system is to provide a fast routability checking. This paper presents the concepts and algorithms to perform this routability checking. The proposed routability checking runs more faster existing methods, by means of decomposing layout region into several pieces. Its efficiency is shown by experimental results.

1 まえがき

現在，プリント基板やMCM (Multi Chip Module) の詳細配線においては迷路法や線分探索法といった逐次配線手法が広く普及している．逐次配線の問題点の一つとして，既配線障壁によりそれ以降の配線が不可能となる場合があることが挙げられる．これは通常の逐次配線手法において，既配線のジオメトリ（絶対位置）が確定され，障害物と同様に管理されることに起因する問題であると考えられる [1][2][5]．また近年の高密度化が進んだプリント基板やMCMの設計，特に配線設計，においては扱うデータ量は膨大であり，一度で最終配線パタンを得ることは困難である．このためレイアウトの変更を繰り返し，最終的な配線パタンを得ることが一般的になっているが，ここでも既配線のジオメトリ管理が柔軟な変更を妨げる原因ともなっている．

これに対して，径路トポロジー（相対位置）を管理することにより上記の問題に自然に対処する方法が提案されている [1] [2] [3] [4]．配線径路のトポロジー (スケッチ) を管理する手法では，最終的にトポロジーをジオメトリに変換するために，与えられたスケッチの配線可能性検証と実際のジオメトリを与えるアルゴリズムが必要となる．特に配線可能性検証は，繰り返し行われるレイアウトの変更の度に呼び出されるため高速性が要求される．そこで本稿では特に配線可能性検証に焦点を当て高速な手法を提案する．提案する手法は，従来手法がレイアウト全体に対して行われたのに対し，レイアウト全体をいくつかの領域に分割することによって高速化を目指すものである．提案手法はスケッチレイアウトシステム [5] の1アプリケーションとして用いられており，配線径路のトポロジー表現としてスケッチ表現を用いている．また径路トポロジーを効率的に表現するため基本データ構造として三角形分割を用いている．

以下では，2節でスケッチ，ラバーバンドモデル，三角形分割等の基本的事項に触れ，3節で配線可能性検証の諸定理，4節で提案する配線可能性検証のアルゴリズムについて述べる．そして，5節で計算機実験結果を示し，6節でまとめる．

2 用語の定義

ここでは，準備として基本的な用語を定義する．

2.1 スケッチ表現と三角形分割

スケッチ (sketch) とは，単層のトポロジカルな配線表現のことであり，障害物，端子，ビアなどを表すfeatureと呼ばれる変形不能なオブジェクトの集合Fと変形可能な配線の集合Wから成る．W中のすべての配線は交差しない．配線のトポロジーは，様々なスケッチ表現で表すことができ，文献 [5] で，配線トポロジーを表すデータ表現について考察されている．結果として，多端子ネットの配線トポロジーや配線可能性検証，レイアウトの動的な更新を考慮し，領域を，オブジェクトの境界をその枝して含む三角形に分割（**CDT:Constrained DelaunayTriangulation**[6][7]）し，枝との交差情報により径路を表すのが効果的であることが示された (図1)．そこで，本稿で領域は三角形分割されたものとし，その各枝の交差情報によりスケッチが表されているものとする．

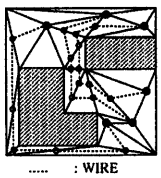

図 1: CDT による配線のトポロジカル表現

2.2 ラバーバンド表現

スケッチは配線のトポロジー表現であり，設計規則の考慮がない．よって，最終的には設計規則を満足する幾何学な配線パタンに変換しなければならない．これには，Leisersonと Maley が定式化したラバーバンド表現を用いた手法が提案されている [8]．**ラバーバンド表現 (rubber-band representation)** とは，トポロジー的に等価である，すなわち互いにホモトピックであるような径路郡のうちそのユークリッド長が最短であるような，その頂点がFに属す多角形径路である．Wの要素が全てラバーバンドであるスケッチをラバーバンドスケッチという．

図2に例を示す．図中 (a) がスケッチであり，(b) がラバーバンドスケッチである．ラバーバンドスケッチでは，まだ設計規則を満足しておらず，F中のオブジェクトに接した状態である．従って，その接点とラバーバンドとの間にスポーク (spoke) という線分を挿入して設計

規則を満足させる．スポークの長さは配線間隔と配線幅により決まり，スポークの数と方向を変化させることによって，任意メトリックの最終配線パタンにすることが可能である (図3)．よって適用する設計規則を変えることにより，様々な配線結果を得ることができる．スポークによってラバーバンド径路間の最小間隔を確保しているラバーバンド表現を**伸長ラバーバンド表現 (extended rubber-band representation)** と呼ぶ (図 2(c))．また，伸長ラバーバンド表現を任意メトリックの最終的な配線パタンへ変換する処理は平面掃引法を用いて行われる（図 2(d))．

これらラバーバンドの柔軟性を生かした実用的な配線システムとして提案された SURF[3][9][10] では，ラバーバンドの動的な更新のための，より効率的なデータ構造やアルゴリズムが提案されている．また，[1] や [2] においてはデータ構造として可視グラフを用いたラバーバンド配線手法が紹介されている．

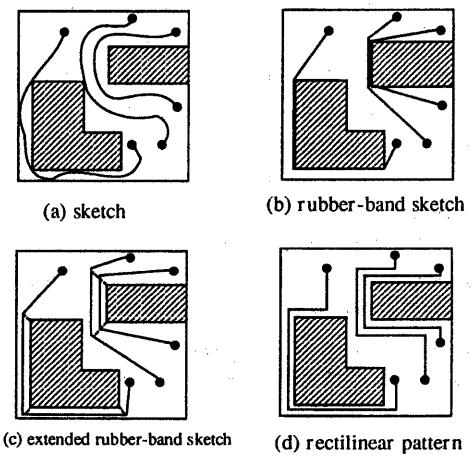

図 2: スケッチ表現とラバーバンド配線

3 配線可能性検証

配線可能性検証は，スケッチが配線可能かどうかを判定する問題であり，次のように定義される [11]．

配線可能性判定問題：与えられたスケッチに対し，設計規則を満足する最終的な配線パタンが存在するか否かを判定する問題

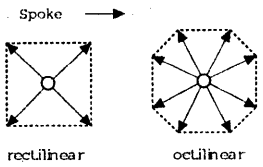

図 3: スポーク

まず，配線可能性判定問題を解く準備としてカットを定義する．

カット 異なるオブジェクトに属する互いに可視な 2 頂点を結ぶ線分．

カットの端点を各々$p(x1, y1)$, $q(x2, y2)$ とし，$d_x = |x_2 - x_1|$, $d_y = |y_2 - y_1|$ とするとき，カットの**容量** $cap(pq)$ は，最終配線パタンによって

rectlinear の場合（水平，垂直線分）

$$cap(pq) = \max(d_x, d_y)$$

octilinear 配線の場合（水平，垂直，45 度線分）

$$cap(pq) = \max(\max(d_x, d_y), \frac{(d_x + d_y)}{\sqrt{2}})$$

で表され，これは pq を通過できる最大の配線幅を意味する．カット pq に交差する径路のために最低限必要な幅をカット pq を横切る配線線分の集合に対しフロー $flow(pq)$ として表す．また，条件

$$flow \leq cap$$

を**容量制約**と呼び，容量制約に違反するカットを**容量違反カット**と呼ぶ．

これらの定義のもと，任意のスケッチが与えられた場合，配線可能性判定問題を解く定理が，まずはじめに，文献 [11] で証明された．

定理 1 全てのカットに容量違反がなければ，そのときに限り配線可能である．

また，文献 [8] では，**クリィティカルカット**と呼ばれる，2 つの障害物間で最小の容量を持つカットを定義することによって，定理 1 から

定理 2 全てのクリティカルカットに容量違反がなければ，そのときに限り配線可能である．

という定理2を導いた．しかし，これらには全てのカット（クリィティカルカット）を表現するためのデータ構造を考える必要があった．

文献[10]では，

定理 3 ラバーバンドスケッチが伸長ラバーバンドスケッチに変換可能ならば，そのときに限り配線可能である．

という定理が示され，特にカット（クリィティカルカット）を表現するデータ構造を考えなくとも，配線可能性検証は，伸長ラバーバンドへ変換処理と同時に行うことができるという効率的な処理が可能となった．

上記の定理に基づく配線可能性検証は，いずれもレイアウト領域全体を一括して考慮するものである．しかし実際には，配線が疎な(配線があまり存在しない)領域も存在する．このため，領域全体を考えなくても，配線が混雑している領域(配線不可能になる可能性がある領域)を求め，領域全体の中で混雑している領域を抽出することができれば，混雑領域内での局所的な配線可能性検証のみで充分である．

そこでまず，混雑している領域を表す統合領域を以下のように定義することによって，定理4を導く．

定義 1 (統合領域) 以下の条件を満たす領域 *reg* を統合領域と呼ぶ．

1. 領域 *reg* は *CDT* の2つ以上の隣接した三角形によって構成される
2. 領域 *reg* 内の全てのスポークが他の領域に押し出さない
3. 領域 *reg* に対し，他の全ての領域内のスポークによる侵入がない
4. 領域 *reg* を構成するどの三角形の集合を除いても上記条件2もしくは条件3に反する（領域 *reg* は他の統合領域を含まない，極小なものである）

定理 4 全ての統合領域内においてラバーバンド径路が伸張ラバーバンド径路に変換できるならばその時に限りスケッチは配線可能である．

証明： 定理3により，配線不可能なカットはスポークにより列挙できる（図4）．統合領域はスポークが伸びる可能性のある隣接領域は必ず統合している．よって定理3より，スポークが安全に伸びることができ，統合領域内の配線可能性が保持されていればスケッチはその時に限り配線可能である．□

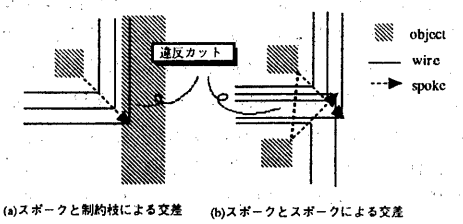

図 4: スポークによる違反カットの抽出

4 配線可能性検証のアルゴリズム

提案するアルゴリズムは定理4に基づいている．領域は三角形分割(CDT)され，径路はそのCDTの枝との交差情報により表現されたスケッチを入力とする．ここでは，特にスケッチ上のすべての径路がラバーバンド表現であるようなラバーバンドスケッチを入力とする．上記のようなデータ構造を用いた場合のスケッチの求めかたについては文献[5]で，またスケッチ表現からラバーバンド表現への変換については文献[8]で紹介されておりここでは特に言及しないことにする．

提案する配線可能性検証のアルゴリズムの概要は，各三角形毎に配線の押し出しを判定し，押し出しのある三角形を統合し統合領域を生成する．そして統合領域内で，スポークを生成することにより伸長ラバーバンドへの変換をすることによって配線可能性を検証するというものである．以下，これらの詳細について述べる．

アルゴリズムの流れ

入力: 径路のラバーバンドスケッチ表現

- 統合領域の生成
- 領域内の配線可能性検証

出力: 違反カット

4.1 領域の統合

この処理によりレイアウト領域全体を統合領域に分割する．CDTの各三角形について，三辺の容量とフローの比較，および三角形内の配線が隣接する領域に影響を与える(配線が隣接三角形に押し出される)か否かの判定を行う．統合領域内部では定理3に基づくスポークによ

る配線可能性検証を行うため（伸長ラバーバンドへの変換），スポークが押し出すと思われる領域の正確な見積もりが必要である．

配線の押し出しは，三角形の各頂点からスポークが伸びる方向の容量とフローを比較することによって判定していく．フローの算出にあたっては，図5のように，一意に定まらない場合があるので，三角形の各辺のフローの内大きなものを求めるべきフローとする．

$$flow(spoke) = \max(flow(ab), flow(ac))$$

そして，以下の条件を満たす時に限り配線が三角形から押し出すとする（図5）．

$$flow(spoke) > cap(spoke)$$

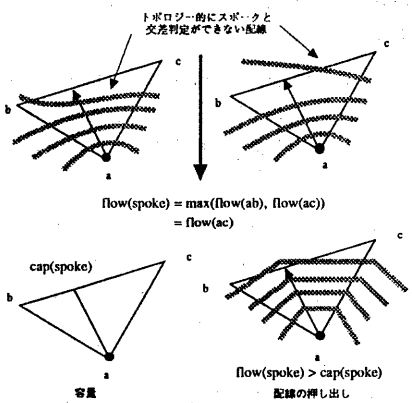

図 5: 配線の押し出しの判定

また，領域が統合された場合の判定方法は，新たに統合される三角形内のフローを $flow(new\_reg)$, $flow(bd \cap cd)$ をカット bd，カット cd をともに横切る配線線分に対するフローとすると，

$$flow(new\_reg) = flow(bd \cap cd)$$

$flow(spoke)$ は，以下のようになり（図6）

$$flow(spoke) = flow(spoke) + flow(new\_reg)$$

また，容量 $cap(spoke)$ は，新たに統合される三角形の容量を $cap(new\_reg)$ とすると，

$$cap(spoke) = cap(spoke) + cap(new\_reg)$$

となり，判定条件に基づき判定する．配線の押し出しがある限り，この操作を繰り返すことによって領域を統合していく．

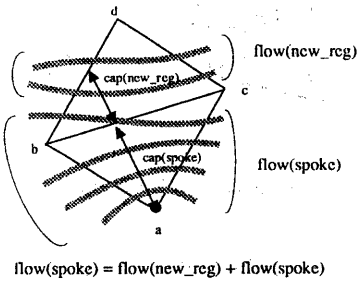

図 6: 統合された場合のフローの算出

また，三角形のある一辺から押し出しがある場合，同一三角形内の他辺からも押し出しがあるならば，その三角形は配線不可能である．これは，三角形内に伸びたスポーク同士が必ず交差するからである（図7）．

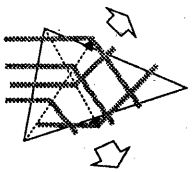

図 7: 2つの押し出し

領域の統合は，各三角形に自らが属する統合領域（統合されていなければ三角形自身）を記憶させておき，新しく三角形を統合する時には，それぞれの三角形に記憶している領域を照合し統合する．統合領域のデータ構造は，三角形の集合をリストでもつことにする（図 8）．これにより一回の領域の統合は定数時間で実現できる．

以上により，領域統合アルゴリズムは次ページのようになる．

4.1.1 計算複雑度

統合領域生成のアルゴリズムは押し出しの判定回数に支配される．押し出しの判定は結局スポークが伸びる方向についてのみ行うことになる．フローの算出には，各三角形内に含まれる配線数に依存した計算時間が必要

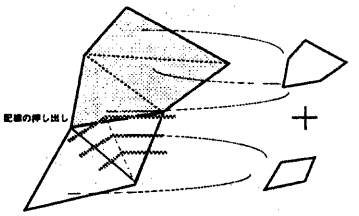

図 8: 領域の統合と統合領域

領域統合アルゴリズム

```
foreach 三角形 tri
  3 辺の配線収容性検証
  foreach 頂点 v
    if (v から三角形内にスポークが伸びる)
      flow = get_flow(tri, v);
      cap  = get_capacity(tri, v);
      if (cap < flow)
        PUSHOUT = TRUE;
        adjacent_tri = get_adjacent_triangle(tri, v);
        merge(tri, adjcent_tri);
        break;
  endfor
  while (PUSHOUT)
    if (adjacent_trig が配線禁止領域)
      break;
    flow = flow + get_adjacentflow(adjcent_tri, v);
    cap = get_capacity(adj_tri, v);
    if (cap < flow)
      adjcent_tri = get_adjacent_triangle(tri, v);
      merge(tri, adjacent_tri);
    else PUSHOUT = FALSE;
  endwhile
endfor
```

である．アルゴリズムの前半部分の 1 つの三角形内に含まれるスポークによる押し出しの比較回数は，スポークの数つまり各節点からは高々2 つのスポークしかつくられないので，結局節点数（feature の端点数）および，フロー算出に要する配線線分数に依存し O(節点数＋配線線分数) である．したがって，入力レイアウトに対し線形である．

アルゴリズムの後半部分の押し出しが複数の領域にまたがる場合の比較回数は，押し出しの回数であり，入力されたスケッチ（配線トポロジー）に依存するが，押し出しが全ての領域にまたがる場合でも三角形分割の三角形数でおさえられる．しかし，押し出しが全ての領域にまたがるといのは実際上は稀なことである．また押し出した領域が配線禁止領域なら，その時点で領域の統合は終了することを考慮すれば，平均的には押し出しの回数は定数回と考えられ，アルゴリズム全体としても O（節点数＋配線線分数）であることが期待できる．

4.1.2 アルゴリズムの正当性

ここでは領域統合アルゴリズムにより求まる統合領域が，定義 1 の統合領域を満たしているかどうかについて検討する．

統合領域の条件は，前記したように

1. 領域 reg は CDT の 2 つ以上の隣接した三角形によって構成される
2. 領域 reg 内の全てのスポークが他の領域に押し出さない
3. 領域 reg に対し，他の全ての領域内のスポークによる侵入がない
4. 領域 reg を構成するどの三角形の集合を除いても上記条件 2 もしくは条件 3 に反する（領域 reg は他の統合領域を含まない，極小なものである）

である．

まず 1 について考える．1 つの三角形のみから成る領域は，配線が通っていても押し出しがなく，他の領域からの配線の押し出しもない領域であり，統合領域には含まれない．このような，三角形内では配線可能であることは明らかである．

2 について，上述した押し出しの判定は，図 5, 6で示したように，領域内の配線のうちスポークにかかる可能性のある配線すべて（実際にはかかることがなくても）を考慮している．よってスポークの存在する三角形内の配線についての押し出しは全て判定できる．

条件 3 については，条件 2 によりスポークが押し出す領域は全て統合することが保証されているため明らかである．

条件 4 については，生成された統合領域は極小のものではないので，この条件を満たすものではない．しかし定義による統合領域を全て含んでいる．したがって，必要とは限らないが十分な領域を統合しているので，配線可能性検証を行う上においては問題がない．

4.2 統合領域内の配線可能性検証

定理 3 に基づいて，統合領域内にスポークと呼ばれる各オブジェクト間の最小間隔制約および各配線線分の

配線幅を考慮した線分を生成し配線可能性を検証する．

まず，各節点（featureの端点）から障害物との最小配線間隔を考慮したスポークを生成しておく．ただし，スポークはoctilinear配線への変換のため凹節点から2方向へ1本ずつ伸ばす．次に，各節点に接するラバーバンド径路の数に応じてスポークを生成し，径路をスポークに沿って1本づつ伸張する．この際に伸長した径路が他のスポークや節点に交差しないかを領域探索を行い判定して，交差するものについてはその節点を待ち行列に入れ，後で同様な処理をする．これらの処理の流れを図9に，そのアルゴリズムの詳細を以下に示す．

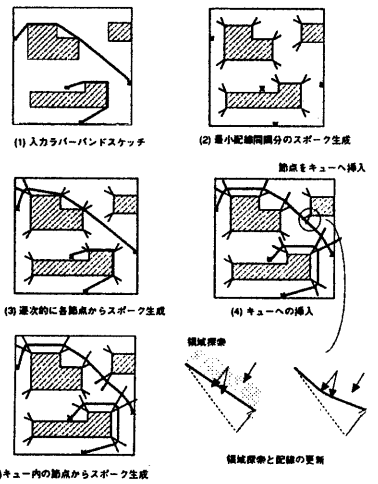

図 9: スポークの生成

ラバーバンドから伸長ラバーバンドへの変換アルゴリズムについては，文献[10]で紹介されているが，提案するアルゴリズムでは，領域探索を効率的に行うことができるよう，各featureから最小配線間隔を考慮したスポークを生成している．これにより，領域探索は，線分とスポークとの単純な交差判定により実現することができる．また，領域探索のためスポークを三角形に保持しておく必要がある．スポークはCDTの制約枝として管理する方法も考えられるが，提案手法では，レイアウトの変更および更新を考慮し各三角形内に管理している．

5 計算機実験結果と評価

提案する配線収容性検証のアルゴリズムを SUN SPARK station 2(28.5MIPS) 上に C 言語で実装し，計

スポーク生成アルゴリズム

```
入力: 統合領域
出力: 違反カット

領域内の全節点から最小間隔を考慮したスポーク生成
function(v)
  foreach 節点 v
    節点 v に接するラバーバンド径路の数のスポーク生成
    foreach 更新された配線線分 w
      w にそって領域探索し w と交差する
          スポークを持つ節点をキューに挿入
    endfor
  endfor

while (!キューが空)
  キューから節点をとりだす v
  function(v)
enfwhile
```

算機実験によって評価した結果を報告する．データは乱数により発生させたものである．比較対象は，従来手法としてレイアウト領域全体に対しスポーク生成アルゴリズムを適用したものを考える．

表1に従来手法と提案手法との処理時間の比較を示す．提案手法の統合領域生成処理はスポーク生成処理の1割程度の時間で実現されており，全処理時間に対して高速に動作することが確認できる．

また，従来手法に比べ，いずれのデータについても，処理時間の短縮が確認（従来手法の23.3%～96.1%の処理時間で実現）でき，提案手法の有効性が示されたといえる．入力レイアウトによっては，統合領域が非常に小さくなり，最高で処理時間が1/4まで短縮されている．これは，入力レイアウトに依存するところが大きいが，配線が偏っている場合など各統合領域自体が小さくなればなるほど高速化が期待できることを示している．

6 おわりに

提案手法は独立な統合領域内部での配線可能性検証を行うが，統合領域を求める処理は容量とフローの計算であり手間は非常に小さくなるため，従来手法に比べ高速化が期待できる．特に混雑度が均一でない場合など，生成される統合領域が少ない程，高速化が期待できることを実験的に確かめた．また，本手法はスケッチ配線後に素子やビアの移動を行う際に局所的に配線可能性検証を行うのにも有用である．

表 1: 実験結果

データ		従来手法	提案手法			
節点数 A	ネット数	処理時間 a (sec.)	全統合領域の節点数 B (B/A%)	処理時間 (sec.) 統合領域生成	スポーク生成	合計 b (b/a%)
320	20	2.567	168 (52.5)	0.400	2.067	2.467 (96.1)
320	20	3.117	171 (53.4)	0.367	2.333	2.700 (87.0)
320	30	8.016	246 (76.9)	0.533	6.783	7.216 (90.0)
1280	30	7.700	422 (33.0)	1.217	5.450	6.666 (86.6)
840	50	26.730	344 (41.0)	0.716	9.866	10.58 (39.6)
2760	50	33.230	320 (11.6)	2.267	5.466	7.750 (23.3)
440	100	5.883	288 (65.5)	0.316	4.333	4.650 (79.0)
560	100	97.980	473 (84.5)	0.684	84.850	85.534 (87.3)

謝辞

本研究は，文部省科学研究費補助金：奨励研究 (A)06855045（平成 6 年度）「柔軟性の高い LSI レイアウト設計手法に関する研究」の援助のもとに行われたものである．

参考文献

[1] 粟島亨，佐藤政生，大附辰夫: " ラバーバンド表現に基づいた逐次配線手法，" 信学春季全大 A-103 (1991).

[2] 粟島亨，田中博，福井省三，佐藤政生，大附辰夫: " ラバーバンドモデルに基づいた逐次配線手法の実装，" 信学技法 VLD92-39， pp.25-30 (1992).

[3] W.W-M.Dai, T.Dayan and D.Stacpelacre: " Topological Routing in SURF : Generating a Rubber-Band Sketch," *28th DA Conf.*, pp.39-44 (1991).

[4] 村田，梶谷:" アナログレイアウトエディタに適した位相配線のデータ構造と修正アルゴリズム，" 情処研報 DA67-12, pp.85-92 (1993).

[5] 田中博， 金沢正博， 田中秀彦， 佐藤政生， 大附辰夫: " スケッチ表現に基づく多層配線システム， " 情報処理学会研究報告 DA70-9, pp.63-70 (1994).

[6] L.P. Chew:"Constrained Delaunay Triangulations," *Algorithmica*, pp.137-151 (1987).

[7] Y.Lu and W.W-M.Dai: " A Numerical Stable Algorithm for Constructing Constrained Delaunay Triangulation and Application to Multichip Module Layout," *International Conference on Circuits and Systems*, pp.644-647 (1991).

[8] C.E.Leiserson and F.M.Maley: " Algorithms for Routing and Testing Routability of Planar VLSI Layouts," *Proc. STOC*, pp.69-78 (1985).

[9] W.W-M.Dai, R.Kong, J.Jue and M.Sato: " Rubber Band Routing and Dynamic Data Representation," *Proc. ICCAD*, pp.52-55 (1990).

[10] W.W-M.Dai, R.Kong, and M.Sato: " Routability of Rubber-Band Sketch," *28th DA Conf*, pp.45-48 (1991).

[11] R.Cole and A.Siegel: " River routing every which way, but loose," *Proceeding of 25th Annual Sumposium on Foundations of Computer Science*, pp.65-73 (1984).