LSI レイアウトにおけるポリゴン配線の通常配線変換

江 達夫 （松下システムテクノ (株)）

渡邊 孝博 （知能情報システム工学科）

Extraction of Routing-Trees from Polygon Routing-Patterns in an LSI Layout

Tatsuo GO （Matsushita Systems & Technology Co.LTD）

Takahiro WATANABE （Department of Computer Science and Systems Engineering）

In LSI layout design, two kinds of wire patterns are used to connect equipotential terminals. One is a so-called "ordinary rouitng-pattern" which is defined by line segments with a given width along a route, and another is a "polygon routing-pattern" whose shape is defined by coordinates of polygon's vertices.

The latter is difficult to deal with layout CAD/DA tools like a layout compactor, because direction of wire's expansion and contraction cannot be recognized.

So, we propose an algorithm which can efficiently transform a polygon routing-pattern into an ordinary one, by constructing a routing tree connecting terminals in each polygon pattern.

Experimental results show that a transformed routing pattern obtained by the proposed algorithm is good for existent CAD tools.

Key Words: *LSI, Layout,Compaction, Polygon Routing, Steiner Tree, Maze Method*

1 はじめに

近年，コンピュータや通信分野などの電子機器は急激な発展を遂げており，それらの要素部品である LSI は，開発品種数の増大と設計期間短縮の問題に直面している．特に，LSI の集積度があがるにつれて，より高密度なレイアウトが求められており，LSI 設計の自動化は極めて重要な課題となっている．

高密度なレイアウトを実現する技法に，素子の配置と素子間の配線を決定したあとに実行されるコンパクション処理[1)]がある．そこでは，配線結果である接続関係を維持しながら配線長さを伸縮可能とし，配置配線図形を移動してレイアウト面積を縮小することが試みられる．通常の配線図形は中心線の経路座標と配線幅からなるデータで与えられるので, 配線の長手方向 (伸縮可能な方向) と幅方向が自明であり，コンパクション処理が容易に行える．しかし，図-1(a) のような配線図形の外形座標で定義されるポリゴン配線は，長手方向が判別できないために，コンパクション処理が行えない．LSI レイアウトにはポリゴン配線と通常配線が混在している場合があり，ポリゴン配線がコンパクション処理の障害となっている．

そこで本研究では，ポリゴン配線に対してもコンパクションを可能とするために，ポリゴン配線から長手方向と幅方向を決定する木構造 (図-1(b)) を抽出し，通常配線に変換する手法を提案する．

2 準備

(1) コンパクション処理

コンパクション処理とは，配置配線結果を図形集合とみなして，接続関係を維持し，配線幅と配線間スペース

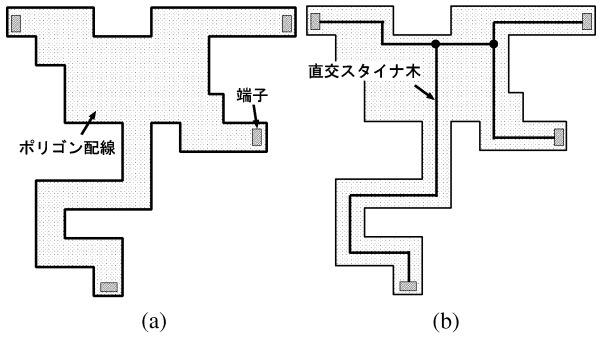

図-1: ポリゴン配線と木構造 (直交スタイナ木)

を考慮しながら，素子（セル）を移動して詰めていくことにより，空き領域を削減し，レイアウト面積を縮小する処理である (図-2)．この処理のために，通常, 配線の長手方向に伸縮可能とする．

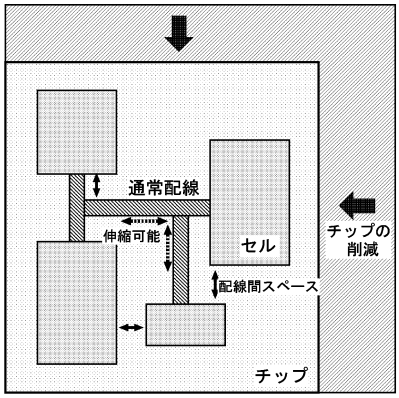

図-2: コンパクション処理

(2) ポリゴン配線と通常配線

図-3 は，形状は同じである二通りの配線表現である．CAD データベース内においては，通常配線の経路は中心線の折れ曲がり点の各座標データと幅データで構成され，中心線に一定の幅を持たせることによって幅つきの配線と成る．一方，ポリゴン配線は周囲の各座標データで構成され，多角形内部を配線領域とする．図-4 は，通常配線 (wire) とポリゴン配線 (polygon) データの記述フォーマットを示している．また，ポリゴン配線および通常配線が混在しているレイアウトの様子を図-5 に示す．

このようにポリゴン配線は，通常配線と幾何学的データ表現が異なり，配線図形を形作る長手方向 (伸縮可能方向) を認識できない．

一般に，自動配線プログラムでは通常配線だけで配線が行われるが，回路規模の小さいものを人手設計するときや，配線修正を対話型 CAD を用いて行う際に，ポリゴン配線が利用される．なお，ポリゴン配線は，チップレベルではなくトランジスタレベルで主に発生する．

(3) スタイナ木

スタイナ木とは，端子の集合が与えられて，端子以外に新たに分岐点となるノード (スタイナ点）を付け加えてもよいという条件のもとで，端子および分岐点の全てを繋げた木である．スタイナ木のうち，枝の長さの総和が最小なものを最小スタイナ木という．垂直・水平枝だけからなるスタイナ木を直線スタイナ木 (RST:Rectilinear Steiner Tree) という．LSI 設計では水平・垂直な配線が行われ，しかも，配線長総和最小化が望まれるので，配線設計は最小 RST を発見する問題と同等になる．

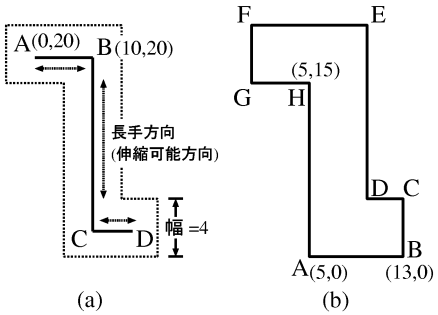

図-3: 通常配線とポリゴン配線

```
wire  width = 4   0,20  10,20  10,0  15,0
polygon  5,0  13,0  13,5  10,5  10,20  0,20  0,15  5,15
```

図-4: 記述フォーマット

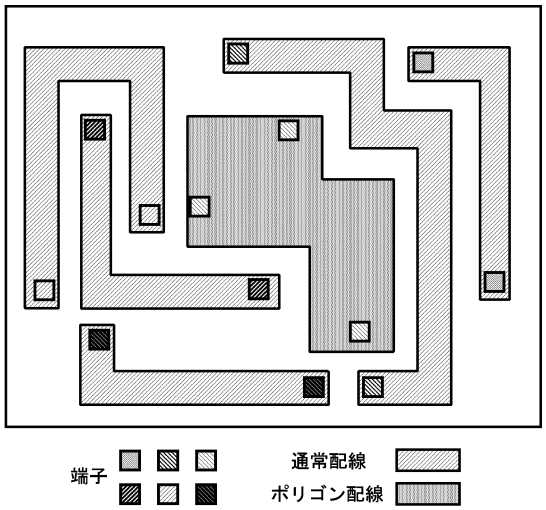

図-5: ポリゴン配線および通常配線の混在レイアウト

3 ポリゴン配線の通常配線変換

通常は，ポリゴン配線と通常配線が混在しているが，ここではポリゴン配線に議論を絞る．本研究では既存のコンパクション・アルゴリズムを利用するために，ポリゴン配線の長手方向と幅方向を決定する木構造の内，枝の総距離が最小に近い木構造を作成し，通常配線に変換する．この結果，ポリゴン配線を含む LSI レイアウト・コンパクション問題を通常のコンパクション問題として解くことができる．

ポリゴン配線の長手方向と幅方向を決定する木構造を作成する問題は「ポリゴン内部で端子間の配線を行う最適スタイナ木を見つける問題」に帰着できる．

ここで，スタイナ木としては最小スタイナ木が望ましいが，その構成には処理時間がかかること，および最小でなくとも後続のコンパクション処理によって木形状の縮約が得られるので，直線スタイナ木 (RST:Rectilinear Steiner Tree) を作成することにした．

例として，図-6 では，ポリゴン配線から木構造を抽出して，通常配線だけを用いた配線結果に変換している．このように通常配線として変換できれば，既存のコンパクション・アルゴリズムを用いて，コンパクション処理が行える．

通常配線に変換する処理全体のフローチャートを図-7 に示す．

RST を作成する基本アルゴリズムとして，プリムのアルゴリズムを用いる．しかし，さまざまなポリゴン配線に対して，基本アルゴリズムだけでは解けない場合が発生するために，迷路法[2)]を用いた例外処理を追加している．以下，図に沿って説明する．

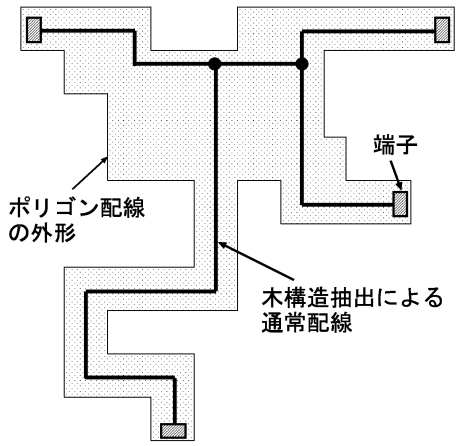

図-6: ポリゴン配線から通常配線へ変換

(1) 前提条件

1. 端子はポリゴン外周上だけに存在するものとする．すなわち，ポリゴン内部には端子は 存在しない．

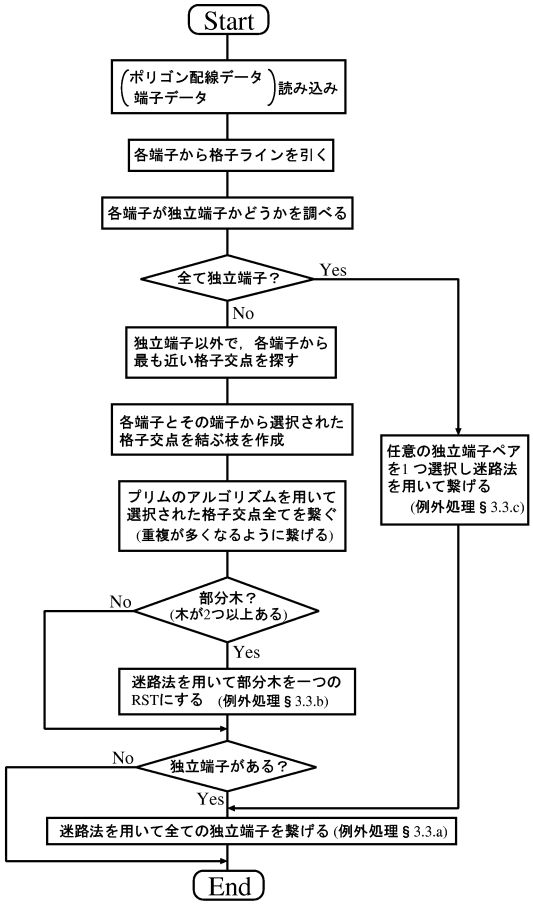

図-7: フローチャート

2. ポリゴンの外形は直交多角形であり，斜め線は含まないものとする．
3. ドーナツパターンのような中抜き部のあるポリゴン配線はないものとする．
4. 枝の距離の評価は，マンハッタン距離を用いる．
5. 端子を繋げる時は，垂直・水平枝だけを用いて繋げる (RST を作成する)．
6. ポリゴン外周上は，配線可能とする．

ただし，前提 1 に関しては，後述する基本アルゴリズム内の「選択された格子交点を繋げる段階」で，内部端子を選択された格子交点とするように拡張すれば解決できる．

また，前提 2 ～ 5 に関しては，現実のレイアウト図形のほとんどが直交線分だけで描かれていること，また，ループ形状の配線を用いる特殊例もあるが，一般には中空部を有する配線パターンは用いられないことから，おおむね合理的な前提である．

前提 6 に関しては，取り出した木構造に均一の幅を与えると，ポリゴン配線領域からはみ出ることになるの

で，実際には，抽出した枝に対して通常配線の幅だけポリゴンの内側にずらす後処理が必要となる．

(2) 基本アルゴリズム

基本アルゴリズムは，次の三つの処理からなる．なお，以降の図中にスタイナ点は示していない．

図-8 のようなポリゴン配線を例に，基本アルゴリズムを説明する．

step1: 全ての端子から外壁に対して垂直 (矢印の方向) に垂直線分を引く．この線分を格子ラインとする (図-8(a))．

格子ラインの交わったところを格子交点とする．

step2: 各端子から最も近い格子交点を選択する．格子交点が重複して選択されることもある（この例では一番左の格子交点）．
そのあと，端子とその端子から選ばれた格子交点を結ぶ枝を作成する (図-8(b))．

step3: プリムのアルゴリズムを用いて，全ての選択された格子交点を繋げる (図-8(c) と図-8(d))．

step3 について説明しておく．図-8(c) で，プリムのアルゴリズムを適用すると，一番左の格子交点を出発点として①から④の順番に枝を作成するが，このとき，斜めに繋がる格子交点は「仮繋ぎ」にしておく．

そのあと，斜めに繋がる枝において (この例では 4 通りの繋げ方がある)，枝の重複部分が多くなるように繋げる (図-8(d))．このようにして，冗長枝の発生を抑制し，結果的に枝の距離総和を小さくしている．

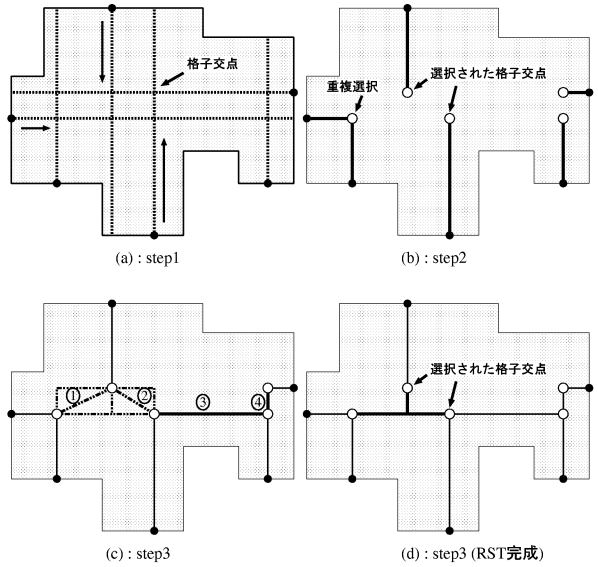

図-8: 基本アルゴリズム

(3) 例外処理

基本アルゴリズムを用いて処理を行うと，どの格子ラインとも交わらない端子 (以下，独立端子と称する) が発生したり，複数の部分木が発生する．このような場合の処理を例外処理と呼ぶ．

これらを解決するために，基本アルゴリズムを適用したあと，各例外処理を適用する．

以下，各例外処理について説明する．

a) 一つの木と複数の独立端子が存在する場合 (図-9)

基本アルゴリズムを適用し格子ラインを引いていくと，木から独立した端子が残る場合がある (図-9(a))．この場合は次のように処理する．

step1: 基本アルゴリズムを適用して，独立端子以外で RST を作成する (図-9(a))．

step2: step1 でできた RST(端子を含む) に，迷路法を用いて全ての独立端子を繋げ (図-9(b))，RST を完成させる．

なお，独立端子が複数個発生する場合，step2 で独立端子を RST に繋げる順番は，現在は，端子データの入力順としている．

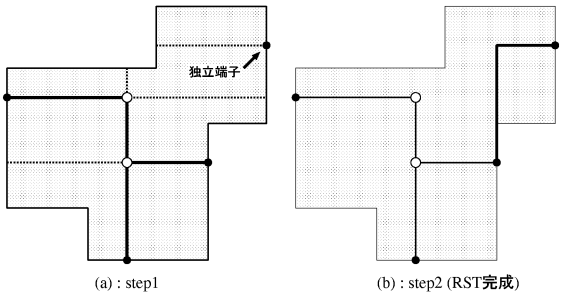

図-9: 一つの木と複数の独立端子が存在する場合

b) 複数の部分木だけが存在する場合 (図-10)

複数の部分木ができる場合 (図-10(a)) は次のように処理する．

step1: 基本アルゴリズムを適用して，各 RST(部分木) を作成する (図-10(a))．

step2: step1 でできた各部分木において，任意の格子交点から自分以外の部分木へ繋がる点を見付けるために迷路法 (前進操作だけ) を用いる．

（図-10(b) では，Ⓢを出発点として前進操作を行っている）．

見かった点を出発点として，再度，迷路法を用いて部分木同士を繋げる枝を作成し (図-10(c))，RST を完成させる．

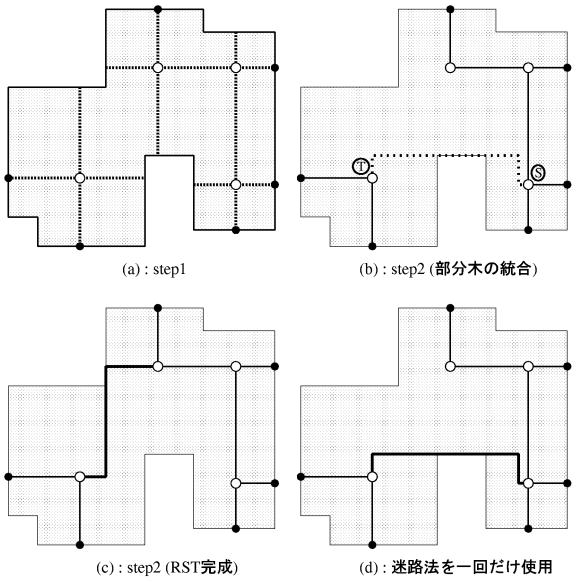

図-10: 複数の部分木だけが存在する場合

なお，部分木が複数個発生する場合，部分木同士を迷路法で繋げる順番は，作成された格子交点の順番とする．

ここで，step2 について補足しておく．部分木同士を繋げるための出発点と目標点をどこにするかが問題となる．理論上は，一方の部分木全体を出発点，他方の部分木全体を目標点として，迷路法を適用すればよい．しかし，部分木の配線上の点を一つずつ出発点として探索することは，処理時間が膨大になる．

そこで，今回は，部分木を作ったときの格子交点を出発点として選んだ．しかし，この方法だと，出発点に選んだ格子交点によって図-10(d) のように冗長枝を作成する場合がでてくる．これを回避するために，迷路法を二回用いることにした．

まず，一回目の迷路法 (前進操作だけ) で最も近い点を見付けて，その点から再度，迷路法を用いて繋げることにした．このようにして，図-10(c) のようにより短い枝を作成するようにしている．

c) 全て独立端子の場合 (図-11)

図-11(a) のように全て独立端子になる場合である．

step1: 任意の端子ペアを迷路法を用いて繋げる．

step2: step1 以降は，迷路法を用いて，順次，端子を既存の RST に繋げるようにしていく．この例では，まず初めに p_1を出発点 p_2を目標点として迷路法を行っている．そのあと，残りの端子を既存の RST に繋げて，RST を完成させている (図-11(b))．

d) 複数の部分木と複数の独立端子が存在する場合 (図-12)

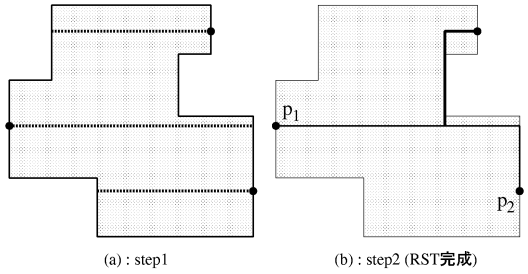

図-11: 全て独立端子の場合

図-12(a) のように複数の部分木と複数の独立端子ができる場合は次のように処理する．

step1: 複数の部分木だけの例外処理と同様に部分木を作成する (図-12(a)(b))．

step2: 部分木を繋げて一つの RST を作成する (図-12(c))．

step3: 一つの木と複数の独立端子の例外処理と同様に，step1 で作成した木に独立端子を繋げ (図-12(d))，RST を完成させる．

ここで，図-12(c) では，step2 の段階で独立端子 p_1が繋がっている．このような場合は，その独立端子は繋がったものとして処理を行う．また，この手法は，他の例外処理においても適用する．

こうすることによって，迷路法を使用する回数を減らし，処理の高速化を図っている．

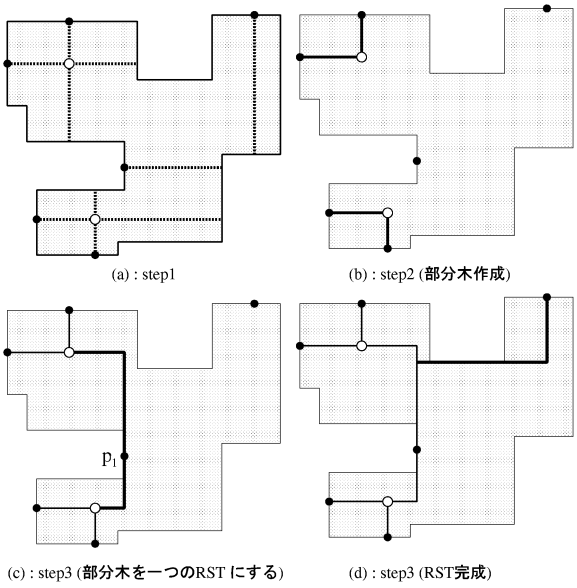

図-12: 複数の部分木と複数の独立端子が存在する場合

4 実験と結果

本手法の有効性を確認するために，基本アルゴリズムおよび各例外処理に対する総配線長，処理時間に関する実験を行った．実験は Intel Pentium II 266MHz (FreeBSD-2.2.8R) を使用し，プログラム言語は C 言語を使用した．

実験に使用したデータは，最大 100×100 *grid* (距離は，座標の 1 を 1 *grid* としている) の平面上に，端子数が異なる 5 種類のポリゴン配線データを基本アルゴリズムおよび各例外処理の実験用に各 5 個ずつ作成した．ただし，一つのポリゴン外壁上に 10 個の端子が存在ような，極端なデータは採用から除外した．

また，基本アルゴリズムと迷路法の比較のため，基本アルゴリズムだけで解けるポリゴン配線データについては，最大 200×200 *grid* の大きさの端子数が異なる 5 種類のポリゴン配線データも各 5 個ずつ用意した．

a) 実験結果

各処理毎の総配線長と処理時間を示す．

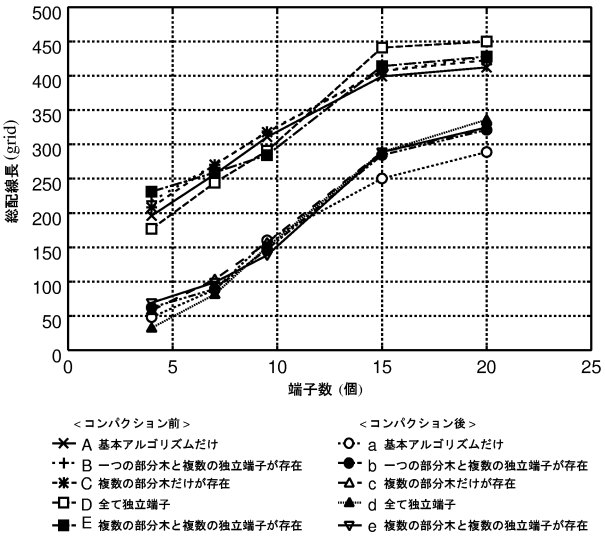

図-13: 総配線長

b) 基本アルゴリズムと迷路法の比較

提案手法では，処理時間の高速化のために基本アルゴリズムではプリムのアルゴリズムから RST を作成する手法を採り，例外処理にだけ迷路法を採用している．

しかし，迷路法は，処理時間がかかる反面「端子間に経路があれば必ず結線できる」という強力な配線手法なので，基本アルゴリズムの部分でも利用することができる．

そこで，我々の提案手法の高速化を評価するために，基本アルゴリズムだけで処理できるポリゴン配線において，基本アルゴリズムで処理を行った場合と全て独立端子とみなし迷路法を適用して処理した場合について実験を行った．図-15 は，総配線長を比較したものである．図-16 はチップ面積を大きく (4 倍) した場合についての処理時間を比較したものである．

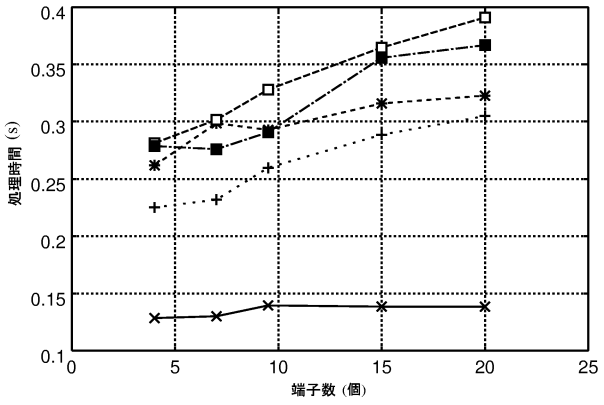

図-14: 処理時間

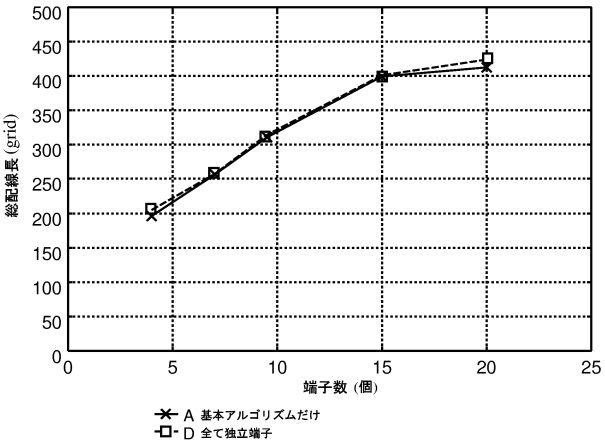

図-15: 総配線長

c) 実行結果

本手法を計算機に実装したときの実行結果を図-17 と図-18 に示す．

d) 考察

全ての処理において，総配線長は端子数が増えるにしたがって削減率が小さくなっている．これは，端子数が増えると，配線数が増えたり配線が複雑になったりするため だと考えられる．

また，総配線長は，基本アルゴリズム，例外処理にかかわらず実用的な配線長となった．

処理時間については，予想どおり「基本アルゴリズムだけ」の処理が一番速い結果となった．また，例外処理

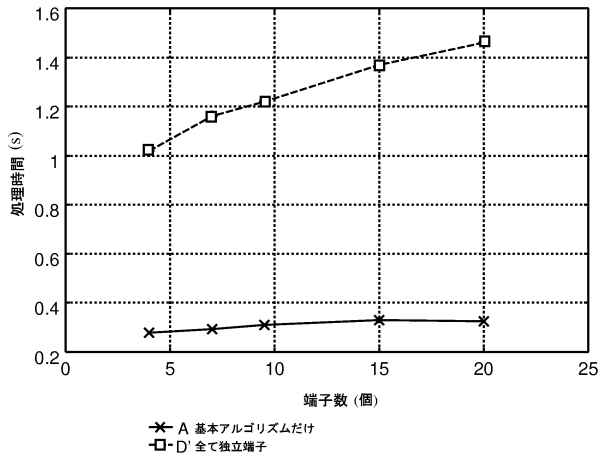

図-16: 処理時間

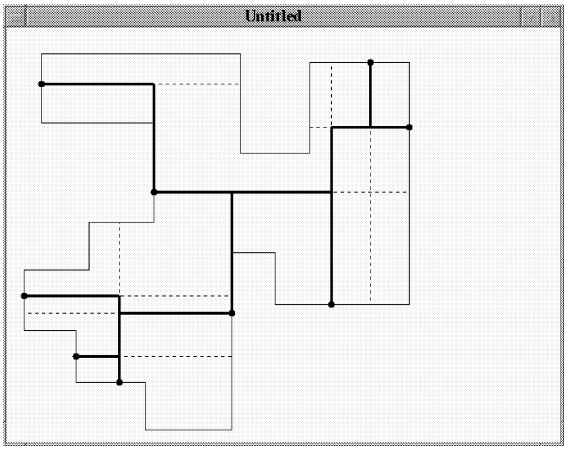

図-18: 複数の部分木と複数の独立端子が存在する場合の実行結果

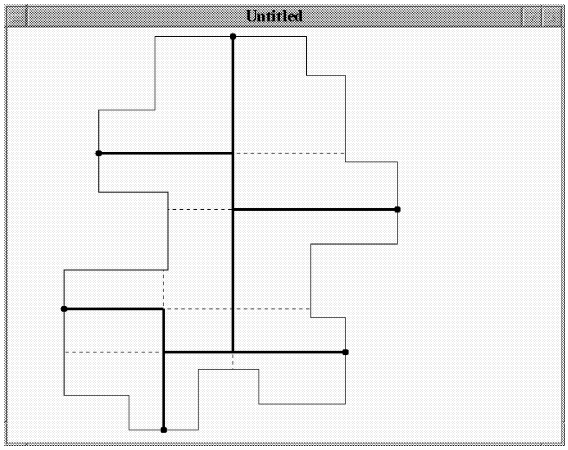

図-17: 基本アルゴリズムの実行結果

では迷路法を使っているので端子数が増えるにしたがって処理時間が増えるが，基本アルゴリズムに関しては，端子数が増えてもほぼ一定の処理時間にな るという結果も得られた．

例外処理の処理時間については，基本アルゴリズムだけの処理時間よりも 1.7 倍 ～ 2.6 倍を要する結果となった．特に，全て独立端子の場合は，全ての端子に迷路法を使うため，端子数が増えるにしたがって，傾きの急なグラフとなっていることが分かる．

また，本手法の高速性を示すために基本アルゴリズムと迷路法の比較をした結果，総配線長に関しては，ほぼ同じ結果となったが，処理時間に関しては，基本アルゴリズ ムの方が速い結果となった．そして，基本アルゴリズムは，迷路法と違ってチップ面積にほとんど影響しないため，端子数が増えるにしたがって差が大きくなった．

5 まとめ

本研究では，ポリゴン配線を含んだ場合でも CAD で統一的に処理できるように，ポリゴン配線データを通常配線データへと変換するアルゴリズムを提案した．すなわち，ポリゴン配線の中に包含されている木構造を抽出，作成し，これを通常配線の中心線として対応付けるものである．しかし，最適な木構造を作成することは実用的な処理時間内では困難である．そこで，我々は発見的手法で近似解を求める方式を採用した．特に，ポリゴン配線がどのような形でも木構造を作成できるように，基本処理と例外処理とに分けて木構造を作るようにしたことが特徴である．

アルゴリズムの評価の結果，総配線長については，基本処理，例外処理にかかわらず，実用的な配線長となった．処理時間については，基本処理の処理時間が一番速く，また迷路法の使用回数が少ない例外処理ほど高速に処理できる結果となった．

また，提案手法と全て迷路法を使用した場合との比較では，総配線長については，全て迷路法を用いた場合と同じぐらいの配線長結果を得られた．一方，処理時間については，例外処理における迷路法使用回数にもよるが，全て迷路法を用いる手法よりも高速に処理できる結果となった．このことから，提案手法は有効であるといえる．

ポリゴン配線内部で木構造を作成するアルゴリズムのためのベンチマークデータは入手が困難なため，総配線長，処理時間については，相対的な割合について示すに止まった．

今後の課題としては，提案したアルゴリズムを土台として，総配線長の点で冗長枝の少ない木構造作成やアルゴリズムの高速化が挙げられる．

参考文献

[1] Naveed A.Sherwani,“Algorithms for VLSI Physical Design Automation, ”Kluwer Academic Publishers, pp.383-407, 1993.

[2] 寺井 秀一,“VLSI デザインオートメーション入門,” コロナ社, pp.128-131, 1999

(2000.8.31 受理)