アルゴリズム 16−11
（1990．7．17）

ジョグ挿入を伴った チップ・コンパクション・アルゴリズム

佐藤政生

拓殖大学工学部情報工学科

山元 渉　中島伸佳+　大附辰夫

早稲田大学理工学部電子通信学科

+ 現在、ソニー㈱厚木テクノロジーセンター

電子回路をいかに小さい面積で設計するかがＬＳＩ設計の鍵である。そのために十数年前より、与えられたレイアウトを一方向に圧縮（コンパクション）する手法に関する研究が活発に行われている。レイアウト面積は配線の折り曲げ（ジョグ）を許してでも小さくすることが好ましい。チャネルに対しては有効なジョグを挿入しながら短時間のうちにコンパクションを行う手法が知られているが、チップ全体に対してはそのような手法は提案されていない。そこで本稿では、上下制約グラフ上で最短径路探索を行うことにより、ジョグ挿入を伴ったチップ・コンパクションを行う高速手法を提案する。また、計算機実験を行った結果を報告する。

A Chip Compaction Algorithm with Jog Insertion

Masao Sato

Department of Information Engineering, Takushoku University

815-1 Tatemachi, Hachioji, Tokyo 193, Japan

Wataru Yamamoto, Nobuyoshi Nakajima+ and Tatsuo Ohtsuki

Department of Electronics and Communication Engineering, Waseda University

3-4-1 Ohkubo, Shinjuku, Tokyo 169, Japan

+ has joined Atsugi Technology Center, Sony Corporation

The one-dimensional compaction is one of the most important tools to design dense LSI chips. To make the chips smaller by the compaction, jog insertion is quite effective. A new fast chip compaction algorithm with automatic jog insertion is presented in this paper. The algorithm is based on Dijkstra's shortest path algorithm on a constraint graph, which is derived from an input layout. Experimental results are also shown.

１．まえがき

ＬＳＩレイアウトの一次元コンパクション手法は、一度に出来るだけ圧縮する一括圧縮と、レイアウトの部分領域を除々に削除していく部分圧縮に大別される。前者に属するものとしては、水平もしくは垂直条件グラフを作成して最長径路探索法を適用する最長径路法[1]、後者としてはコンプレッション・リッジ法[2]が広く知られている。本稿では前者のみを扱い、圧縮は縦方向に行われるものとする。また、入力されるレイアウトは設計規則を満足しているものとする。

一括圧縮手法としては、近年、計算幾何学の分野における手法を適用したものが提案されている[5-10]。特に、チャネル配線領域に対しては拡張平面掃引法[3,4]に基づく優れたコンパクション（スペーシング）手法が提案されている[6-9]。これらの手法の基本思想は、最長径路法と同じであるが、レイアウトを構成する配線線分やビアを下方向もしくは上方向から順次探索してゆけばコンパクションが行われることより、グラフを作成する必要がなく、次のような多くの利点を持つにいたった。

(1) 格子構造によってレイアウトを表現することなく、配線、ビアなどを幾何学的な情報のまま取り扱うので、複雑な設計規則に追従し易い等、柔軟性がある。

(2) グラフを用いないので記憶容量に関するオーバーヘッドが小さい。

(3) アルゴリズムが単純でかつ効率が良い。

(4) 不規則な形状の境界を持つ配線領域を取り扱う。

(5) 多種類の幅を持つ複数の配線を一括して取り扱う。

(6) 自動的にジョグを挿入しレイアウト面積を最小にする。このとき、発生するジョグ数が少ない。

(7) 圧縮幅などの指定が行える。

(8) 左右境界上の端子などの任意のレイアウト要素を固定するよう指示ができる。

(9) ビアのスライディングを行える。

この中で、最長径路法に欠けているものは、(2),(6),(7)であろう。ジョグ挿入が不可能なのは、ジョグ挿入にともなうグラフの節点分離を扱うことが最長径路探索時に困難であるためである。ジョグ挿入がレイアウト面積の最小化に与える影響は大きく、不可欠である[8,9]。

一方、多くの利点を持つ幾何学的手法は、機能ブロックを含むレイアウトを扱うことが不得手であり、チップに対するコンパクションには向いていない。文献[10]では、バッグという概念を用いて幾何学的に処理を行っているが、レイアウトの規模が大きくなると、膨大な処理時間を要している。

そこで本稿では、上記の利点をなるべく継承し、機能ブロックを含むレイアウト全体をジョグの挿入を行いながら圧縮する手法を提案する。さらに、この手法をインプリメントし、計算機実験を行った結果を報告する。提案する手法は文献[11]の手法を拡張したものであり、垂直条件グラフを陰に作成し、この上で最短径路探索を行うことによりコンパクションを行っている。陰にではあるがグラフを作成するので、ジョグの挿入を行いにくいと思われがちであるが、この問題点は、グラフ探索に最長径路法ではなく最短径路法を用いることで解決される。提案手法は、利点(2)は失うが、他の利点に加え次の利点を持つ。

(10) ブロックの取り扱いが可能である。ブロックの形状は四角形に限らず、任意の複合長方形領域でよい。

(11) 端子のスライディングを行える。

(12) 多層レイアウトを扱い易い。

以下では、ビルディングブロック方式ＬＳＩに見られるような機能ブロックを含むレイアウトがｘｙ平面上に与えられるとする。コンパクションはレイアウトをチップ下辺に向けて圧縮することを意味するものとする。配線層数は任意でよい。

２．アルゴリズムの概要

アルゴリズムの説明の前に以下の用語を定義する。

・オブジェクト：スペーサが処理の対象とするレイアウト要素。ブロック、端子、ビア、水平配線を指す。ただし、ブロックは境界の各水平線分がオブジェクトであり、他は水平中心線分が幅を持ったオブジェクトであるものとする。レイアウトの移動はｙ軸方向のみのため垂直配線や境界の垂直線分を陽に考慮する必要はない。

・ベンド：同一層における配線の曲がり点。

・ジョグ：ベンドを結ぶ縦配線線分。

今回提案するチップ・スペーシング・アルゴリズムは大きく分けて３つの段階から構成される。

Phase１．各オブジェクトに上下に隣接するオブジェクトへのポインタを持たせる。この構造はグラフとみることもできるので以後、制約グラフと呼ぶ。

Phase２．制約グラフを探索することによって、オブジェクトを下方に詰める。その際、有効性のありうるジョグをすべて発生させる。

Phase３．Phase２で得られたパタンを指定された幅を越えて拡大（ｙ軸方向に伸張）しない範囲で、オブジェクトを移動させて可能な限りジョグを削除する。

幾何学的な手法は、まずジョグを発生させて下詰めを行ってから、次に不必要なジョグを削除するので、本手法もグラフを作成する点を除けば、本質的に同様の手法である。次節以後に、各Phase に関する処理の詳細を述べる。

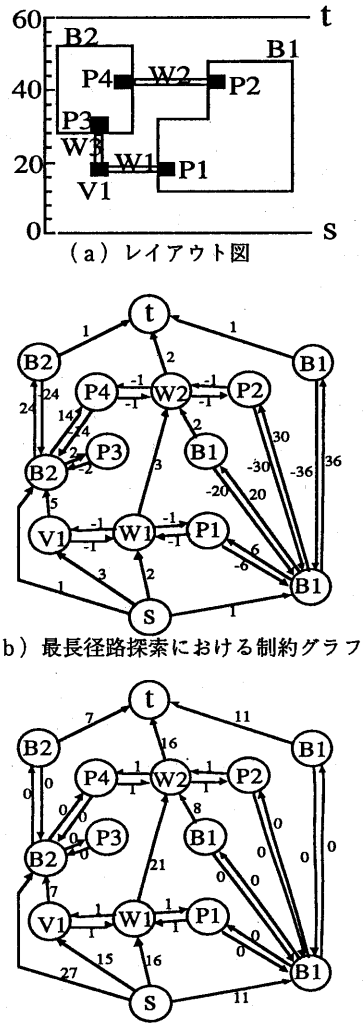
３．制約グラフの作成

制約グラフは、基本的には文献[11]と同様の方法で作成される。同一層上でｘ座標における区間に重なりがあり上下に隣接するか、ｙ座標を共有し左右に隣接する２つのオブジェクト間に枝が設けられる。このとき、同じブロックの上下水平線分のように相対位置に制約があるオブジェクトの間には双方向に有向枝が、配線どうしのように設計規則が許す限り近づくことのできるオブジェクトの間には下から上への有向枝が設けられる。複数の層にまたがるビアやブロック、端子はそれぞれの層で隣接関係を探索することになる。ただし、同じブロックに属する端子間には、ブロックの水平線分と端子の相対位置が固定であることから、端子－ブロック－端子という道から関係が導けるので、冗長を回避するために枝を設けないことにした。この処理は平面掃引法[3]によりＯ（ｎlog ｎ）の手間で実行される。ここで、ｎはオブジェクト数である。図3.1 に例を示す。

次に、この様な制約グラフを作成した理由について述べる。

チャネル・スペーサ[6-9]は、平面掃引法によってレイアウトの下詰め操作、ジョグ削除を行うので、非常に高速である。しかし、平面掃引法を適用するためには、オブジェクトを下方から上方に順序よく探索すれば十分であるという条件が必要である。この条件はブロックの存在するレイアウトには当てはまらない。ブロックはビアと異なり、配線が出る方向が決っており、かつ、複数の配線と接続しているので、ブロックの移動に関しては、集合の概念ならびに矛盾を起こさない探索順序の決定が必要となってくる。

図3.2 において、左のブロックだけを移動させようとするとジョグを挿入する必要がある。しかし、垂直配線の混雑度が２つのブロックの間隔で定義される配線容量を越えてしまうため未結線を生じてしまう。したがって、２つのブロックは同時に移動しなければな



（ａ）レイアウト図

（ｂ）最長径路探索における制約グラフ

（ｃ）最短径路探索における制約グラフ

図３．１　制約グラフ

らない、即ち、２つのブロックを１つの集合として扱わなければならない。図3.3 は、下詰めを行うときに、下方向の境界がない場合にはジョグを挿入して２つのブロックを自由に動かすことができるが、境界の形状によっては配線の下詰め位置がクリティカルになるためにジョグを挿入することができず、２つのブロックの相対位置を変更できない例である。このように、同一集合に属すか否かは、他のオブジェクトの形状に依存する場合がある。図3.4 のレイアウトでは、ブロックB_1,B_2,B_3がジョグ挿入の不可能な配線W_1,W_2によって接続されているので同時に移動しなければならず、オブジェクトの探索をｙ座標の小さい順に行えば良いというわけにはいかない。図中、y_0上のブロックB_1，B_2は、y_1からy_5上のオブジェクトに対する計算が全て終了した後にその位置が決定されることになる。このようなレイアウトに平面掃引法を用いると、走査線がy_0のときにB_1，B_2の仮の位置を決定し掃引を続けなくてはならない。そして、y_5までの探索を行いB_3の位置が決まり、B_1，B_2の位置を変更する必要がでてきた場合には、再びy_1からy_5までの掃引をやり直さなければならない。したがって、掃引する回数がレイアウトの形状に大きく依存してしまうことになる。図3.5 は、２つのブロックの集合の上下関係にサイクルを生じる例であり、これではどちらから探索を行って良いか判断に困る。

以上述べたような状況を一つ一つ検証することに費やされるコストを差し引くと、平面掃引法による手法を用いるメリットはない。ブロックに関わる種々の制約を統一して扱うためには制約グラフを用いるのが最も効率的であると考えられる。

４．下詰め操作

コンパクション・アルゴリズムでよく知られているのは、最長径路探索による手法であるが、本手法では、最短径路探索によってレイアウトの下詰め操作を行う。まず、最短径路探索を適用するための制約グラフの各枝へのコストの与え方や性質を説明し、次に最短径路探索法、また、それを用いた理由について述べる。

４．１　制約グラフの枝のコスト

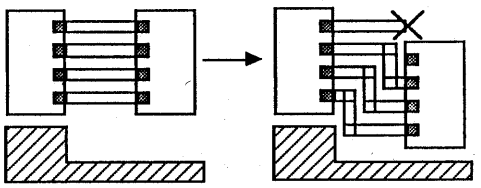


図３．２　配線混雑度とジョグの関係

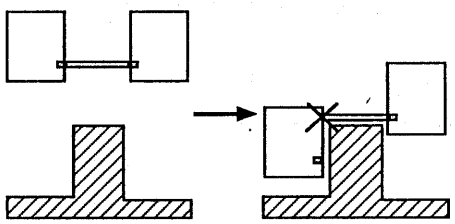


図３．３　レイアウト形状とジョグの関係

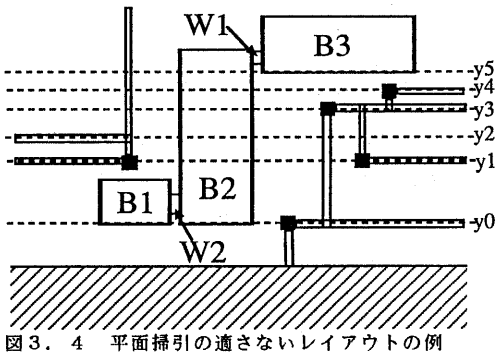


図３．４　平面掃引の適さないレイアウトの例

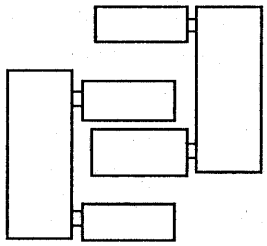


図３．５　ブロック集合のサイクルの例

最長径路探索問題から最短径路探索問題への変換は、今井氏の研究[11]によって証明されている。この変換は単に枝のコストを次式のように書き換えるだけで行われる。

$$New\ d_{uv} = -y_{u0} - d_{uv} + y_{v0}$$

ここで、d_{uv} は節点ｕとｖの垂直条件制約、y_{i0} は節点ｉに対応するオブジェクトの初期ｙ座標値を表す。ここで、初期配置において設計規則を満たしている（最長径路探索において正サイクルが存在しない、最短径路探索において負サイクルが存在しない）ことより、

New $d_{uv} \geq 0$

が成り立つ。この新たな重みの直感的意味は、節点uとvの間に存在する余裕度（vはuにd_{uv}だけ近づける）である。枝の両端点に対応するオブジェクトの種類により枝とコストの関係は3つに分類できる（図3.1(c)参照）。

(1) 相対位置が変化しないオブジェクト対の場合

コスト0の双方向有向枝で対応する節点は結ばれる。各ブロックにおける境界の水平線分対や、水平線分と端子の対がこれにあたる。

(2) 相対位置に上下限の制約があるオブジェクト対の場合

非負コストの双方向有向枝で対応する節点は結ばれる。左右に隣接する配線とビアの対、配線と端子の対がこれにあたる。

(3) 設計規則が許す限り近づくことのできるオブジェクト対の場合

非負コストの下から上へ向かう有向枝で対応する節点は結ばれる。(1),(2) 以外の対がこれにあたる。

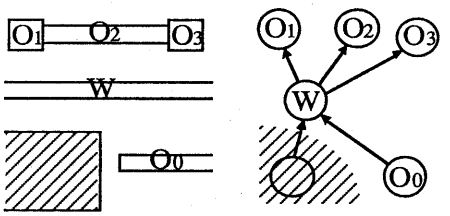
このグラフに対してダイクストラ法[12]を適用すると、各節点には移動すべき量が永久ラベルとして付くことになり、ジョグを伴わずに最長径路法でコンパクションした結果と同じものがO（n log n）の手間で得られる。

4．2 最短径路探索とジョグの挿入

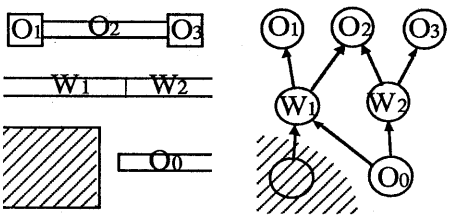
最短経路探索はレイアウトパタンの境界下辺に対応する節点Sをソースとして開始される。Sにはコスト0の永久ラベルが付き、Sに隣接する節点には、その節点への枝と同じコストが仮ラベルとして与えられる。ダイクストラ法による径路探索で各節点に与えられるラベル値は、その節点に対応したオブジェクトが初期の位置よりどれだけ境界下辺に近づけるかという距離を表している。永久ラベルの付いたオブジェクトはレイアウトにおける位置が定まる。位置の定まったオブジェクトによって形成されたレイアウト境界の形状を基に幾何学的な手法[6-9]と同様に配線オブジェクトの分割を行うことができる。分割されたオブジェクト間には何も制約がないので、独立に動くことになり、動く量が違えば、それはジョグを挿入したことと等価になる。仮ラベルを付けるときに配線オブジェクトの分割を行う以外は、通常のダイクストラ法とまったく同じ操作でジョグ挿入を行うことができる。

図4.1にオブジェクトの分割と制約グラフの更新の様子を示す。斜線は探索が終了し位置の確定したオブジェクトを示す。図(a)の配線wはジョグの幅と設計規則を考慮した位置において分割され図(b)のw_a，w_bとなる。制約を表す枝の更新は、分割したオブジェクトに隣接するオブジェクトを既存の枝を通じて探索しながら行うことができる。

分割の対象となる部分はラベルの未確定な、即ち、未探索な部分である。したがって、オブジェクトの分割に伴うグラフの節点分離はグラフの未探索な部分で行われることになるため、グラフの探索をやり直すことはない。つまり、レイアウト要素を一通り見ていくだけでジョグを発生させながらレイアウトの下詰めが完了するのである。これより、ジョグ挿入を伴った下詰め操作の計算複雑度はO（N log N）となる。ここで、Nは分割した結果として生じた総オブジェクト数である。



（a）オブジェクト分割前のレイアウトと制約グラフ



（b）分割後のレイアウトと制約グラフ

図4．1 配線オブジェクトの分割

4．3 最短径路探索の利点

最短径路探索を用いることによって生ずる利点をまとめる。

(1) ジョグ挿入を行わないとしても、最長径路法では一般的にO（n^2）の手間を要するのに対し、O（n log n）の手間で処理を行う。

(2) 文献[10]では、相対位置を変えることのできないオブジェクトの集合をバッグに入れていた。1つバッグに属するオブジェクトは、最短径路探索ではすべて同じ値の永久ラベルが与えられることになる。したがって、バッグの取り扱いにともなう複雑な操作を行わずに、バッグの働きと等価なことを自然に行っている。たとえば、図3.2 の場合には端子のスライディングをないとすると、図4.2 のように左のブロックから端子、配線、右のブロックの端子、右のブロックと同じ値のラベルの伝達が行われるので、左右のブロックには同じ値の永久ラベルが付く。図中、斜線は永久ラベルの付いたオブジェクトを表す。

(3) 移動可能距離が小さいオブジェクトほど早く永久ラベルが付くということは、その他のオブジェクトは永久ラベルの付いたオブジェクトよりも移動可能距離が大きい、つまり、自由度が高いことを示す。この性質より、オブジェクトに永久ラベルが付く順序がそのままオブジェクトの位置を決めるべき順序となり、図3.3 から図3.5 の問題点がすべて解決される。

(4) ジョグ挿入が容易に行える。

5．不要なジョグの削除

可能な限りのジョグを発生させながら下詰めを行うことにより、レイアウトの最小ピッチが決定される。しかし、この時点では不要なジョグが多く存在する。不要なジョグは次に水平方向にレイアウトを圧縮する際に妨げとなる。また、配線抵抗の増加や歩留まりの低下をもたらす結果にもつながる。そこで、レイアウトをより最適なものとするため不要なジョグの削除操作を行う。ジョグ削除操作は基本的には、文献[9] と同様であるが、ブロックを下詰め位置固定にした場合と移動可能にした場合の2通りがある。

5．1 ブロック位置を固定にしたジョグ削除

チップの一部に関してのみコンパクションを行い、空き領域を作成し、そして、その空き領域に別の回路を持って来るような場合には、下詰め操作後のブロックの位置を固定しておくことが望ましい。

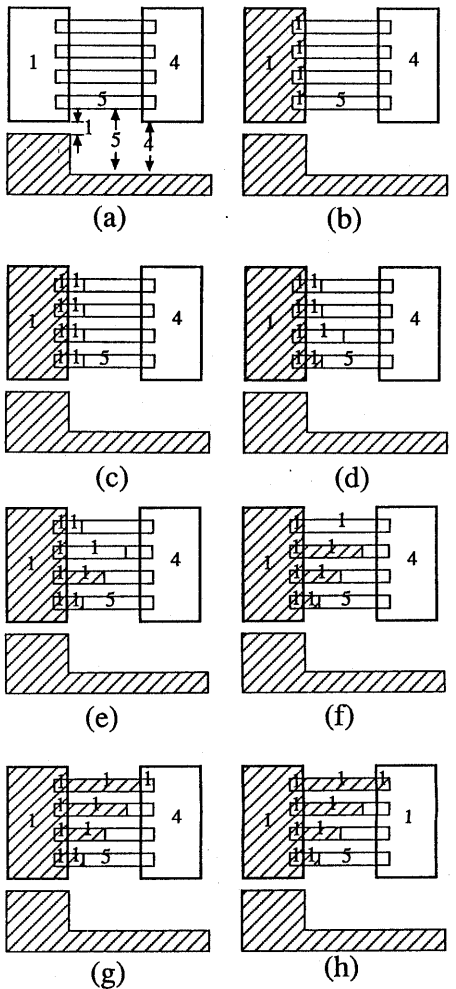


図4．2 永久ラベルの伝達

ブロックを固定した場合に移動可能なのは、配線オブジェクトとビアオブジェクトだけである。したがって、文献[9] の手法が適用できる。これは、上方のオブジェクトから順に探索してゆき、そのオブジェクトを移動させることによってジョグ削除が行えるならば、つまり、隣合う同一ネットに属する水平配線オブジェクトを同一y座標に移動させ合体させることができるならば、その位置に移動させる。そうでなければ、許される範囲でなるべく上方に移動させる。この操作は明かにO（N）の手間で実行される。

5．2 ブロック移動を伴うジョグ削除

ブロックを上方に移動させると不必要なジョグをさ

らに減少させる可能性がでてくる。ブロックの移動がジョグ削除に関して今までと違うことは、矛盾を生じないオブジェクト移動順序を求めることと、ブロックが複数の端子を持つので、相対位置を変えない複数のオブジェクトに関して、ジョグ削除を総合的に考えなくてはならないことである。

オブジェクトの探索順序は、下詰め操作と同様に、最短径路探索法によって定まる順序を採用する。このために、下詰め操作終了時の制約グラフの枝の向きを変え、枝のコストの計算を行う。この操作はO（N）の手間で行なわれる。

ジョグ数の削減は、図5.1 のように、同じy座標値を持つ端子の組数を増やすことによって行われる。したがって、隣接ブロックの端子位置のマッチング問題を解くことになる。図5.2 はこの問題をモデル化したものである。議論を簡単にするために、ここでは2つのブロックX，Yだけがあり、各ネットは2点間を結ぶものとする。各端子は自分が属するネット名を持ち、同じネット名を持つ端子は結線されなければならない。ブロックX上の端子i(i=1-5)のブロック下辺からの距離をそれぞれx_iとして2次元座標のx座標とし、ブロックYについても同様にすると図(b) のように各ネットに対し点が描ける。このグラフ上において傾き1の直線$y=x+\delta$と交差する点（x_i, y_i）(i=1-5)の数を計算する。すると、ブロックYの下辺がブロックXの下辺よりもδだけ低ければ、何組の端子のy座標がマッチングするかがわかる。最大マッチングを求める計算は、単にy_i-x_iの値をソーティングし、最も同じ値の多いものを選ぶことで行われるのでO（p log p）の手間で実行される。pはブロックの端子の数である。

オブジェクトは最短径路探索時に、自分の仮ラベルが永久ラベルになるときに位置が確定する。各オブジェクトは置くことが可能な範囲を持つ。それは、そのオブジェクトの下部で永久ラベルを持つものがあれば、その最大値から今自分が持っている仮ラベルまでの範囲である。上記の最大マッチングもこの範囲において求めなければならない。通常の最短径路探索では、仮ラベルより小さいラベルを永久ラベルとすることはない。これは、通常の場合には、枝のコストが最低必要となるもの（下限）を表すのに対し、コンパクションでは、最大移動可能距離という上限を表すからである。

上述の最大マッチング・アルゴリズムを端子と配線のスライディングを許したモデルを扱えるよう拡張する。端子に対して、図5.3(a)のように、最も上に移動したときの配線の中心線のy座標をx_{ih}（y_{ih}）、最も下のy座標をx_{il}（y_{il}）とする。すると、対応するグラフ上では図(b)のように四角形が描かれる。ここで、傾き1の直線と交わる四角形の最大数が最大マッチングとなる。同図では、2つの直線の間であれば5つのネットがマッチングすることを示している。このモデルにおいて最大マッチングを求めることは、斜め線による平面掃引法を用いることによりO（N log N）の手間で行うことができるので、単純なモデルと同じ複雑度となる。

以上の例では2点間ネットを考えていたが、左に2端子、右に3端子といった多点間ネットの場合には、6つの点を設けることによって同様に扱うことができる。しかし、実際にはこのような例はほとんどない。

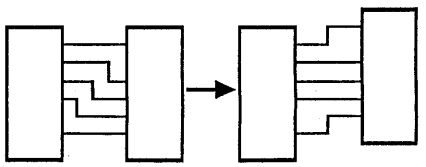
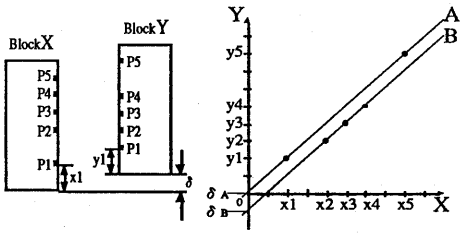
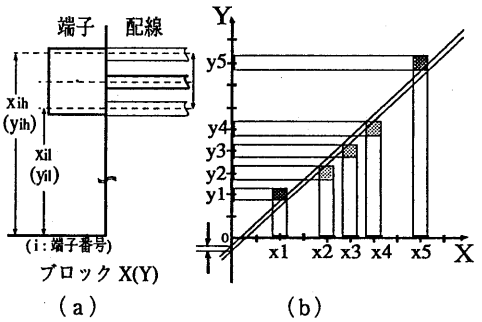


図5．1　ブロックの相対位置とジョグ数の関係



（a）　（b）

図5．2　端子位置のマッチング



（a）　（b）

図5．3　配線－端子スライディングを考慮する場合

６．計算機実験結果

図6.1 から図6.4 に実験結果を示す。実験はSUN4/110上で行った。図6.3 は下詰め後ブロックを固定した結果で1.19[sec]の処理時間を要し、ジョグ数は４２となった。図6.4 はジョグ削除のためにブロックを移動した結果で処理時間1.65[sec]、ジョグ数１３である。また、オブジェクト数は入力時１９６、下詰め操作後３００であった。

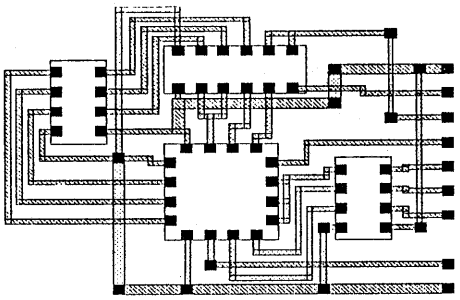


図６．１　入力レイアウト

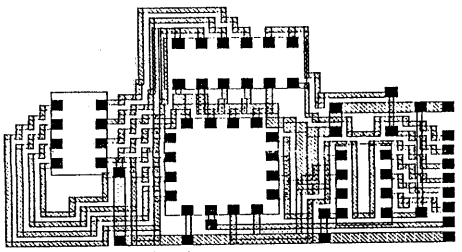


図６．２　下詰め操作後のレイアウト

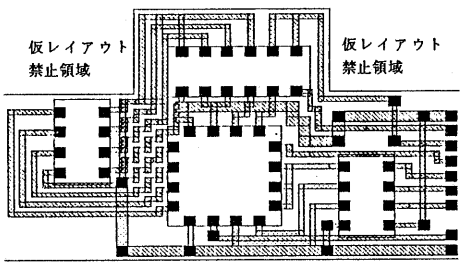


図６．３　ジョグ削除操作後（ブロック固定）

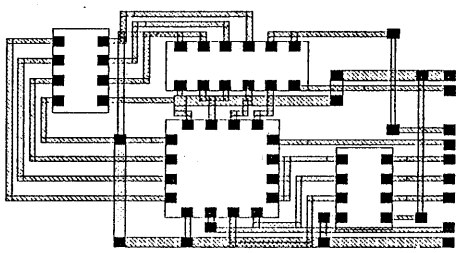


図６．４　ジョグ削除操作後（ブロック移動）

７．あとがき

本稿では、ブロックを含んだレイアウトのコンパクション・アルゴリズムを提案した。このアルゴリズムの計算複雑度はＯ（Ｎlog Ｎ）となり、通常、Ｎ<<ｎ^{2}である。ここで、Ｎは下詰め操作後のオブジェクト総数、ｎは入力されたオブジェクト数である。今後の課題としては、実回路データでの追加実験による提案手法の評価があげられる。

謝辞　本研究の一部は財団法人安藤研究所奨励金の援助のもとに行われたものである。

文　献

[1] Y.E.Chao, A.J.Korenjak, and D.E.Stockton, "FLOSS: An Approach to Automated Layout for High-Volume Design," Proc. 14th DA Conf., pp.138-141, 1977.

[2] S.B.Akers, J.M.Geyer, and D.L.Roberts, "IC Mask Layout with a Single Conductor Layer," Proc. 7th DA Workshop, pp.7-16, 1970.

[3] D.T.Lee and F.P.Preparata, "Computational Geometry --- a Survey," IEEE Trans. on Computers, Vol.C-33, No.12, pp.1072-1101, 1984.

[4] 佐藤，大附，"Enhanced Plane-Sweep Methods for LSI Pattern Design Problems," 信学技報，CAS86-199, pp.87-94, Jan.1987.

[5] 鎌田，佐藤，大附，"多層レイアウトパタンに対する高機能コンパクション手法，" 信学技報，CAS86-55, pp.39-45, May 1986.

[6] J.Royle, M.Palczewski, H.VerHeyen, N. Naccache, and J.Soukup, "Geometical Compaction in One Dimension for Channel Routing," Proc. 24th DA Conf., pp.140-145, 1987.

[7] X.-M.Xiong and E.S.Kuh, "Nutcracker: An Efficient and Intelligent Channel Spacer," Proc. 24th DA Conf., pp.298-304, 1987.

[8] C.-K.Cheng and D.N.Deutsch,"Improved Channel Routing by Via Minimization and Shifting," Proc. 25th DA Conf., pp.677-680, 1988.

[9] 金，中島，佐藤，大附，"ビア削除を伴った高速多機能チャネルスペーサ，" 電子情報通信学会論文誌Ａ分冊，Vol.J72-A, No.2, pp.349-358, Feb. 1989.

[10] X.-M.Xiong and E.S.Kuh, "Geometric Compaction of Building-Block Layout," CICC'89, May 1989.

[11] H.Imai, "Notes on the One-Dimensional Compaction Problem of LSI Layouts Viewed from Network Flow Theory and Algorithms," Trans. IECE Japan, Vol.E69, No.10, Oct., 1986.

[12] E.W.Dijkstra,"A Note on Two Problem in Connexion with Graphs", Numerische Mathematik, 1, pp.269-271, 1959.