設計自動化 36－7
（1987. 2. 17）

ビィルディングブロック方式 ＬＳＩ用レイアウトコンパクタ

灘岡 満　安藤 宏
（沖電気工業株式会社）

本コンパクタはＶＬＳＩレイアウトシステム（ＶＩＬＬＡ）の一サブシステムであり、対象は自動配置／配線プログラムによって設計されたレイアウトである。コンパクションは３つのステップより成る。最初にレイアウト要素を頂点、レイアウト要素間の制約関係（例えば最小間隔以上離れねばならない）を枝とする制約グラフを作成する。次にこの制約グラフを用いて配線の自動折曲げも行なってｘ／ｙどちらか一方に可能な限り詰めたレイアウトを得る。最後に不要な配線折曲げの除去や線長の低減を行ないレイアウトを改善する。

本稿ではこのコンパクションの手法と実行結果について述べる。

A Layout Compactor for Building Block LSI

(in Japanese)

by Mitsuru NADAOKA and Hiroshi ANDO
(OKI Electric Industry, Co.Ltd, Hachioji-shi, Tokyo 193 Japan)

This compactor is a subsystem of VLSI layout design system called VILLA and compresses the layout block designed by automatic placer and router. This compaction consists of three steps: first, to build the constraint graph that vertices corresponds to layout elements and edges corresponds to constraints between layout elements (for example, the elements must separate more than minimum space), second, to minimize block area 1-dimensionaly using automatic bending wire by solving constraint graph, and last, to improve the layout that is removing unnecessary wire bends and shortening wires.

This paper describes compaction method and its experemental results.

１．はじめに

本コンパクタは、ビィルディングブロック方式ＬＳＩ用の自動レイアウトシステム(VILLA)[1]の一サブシステムであり、自動配置、自動配線の次に位置する。

この自動配線は、手法としてラインサーチ法を用いている。ラインサーチ法は自由度の大きいレイアウトモデルを取扱えるという利点を持つが、配線チャネルの大きさが固定あるいはグリッドを用いるという欠点を持っている。配線チャンネルの大きさが固定であるため設計しようとするブロックの面積を小さくするには、高精度で配線チャンネルの大きさを見積もらねばならない。なぜなら、配線チャンネルの大きさが大き過ぎれば設計しようとするブロックの面積が増大するし、小さ過ぎれば未結線が多く発生してしまうからである。しかし、この配線チャンネルの大きさを高精度で見積るのはなかなかむずかしい問題であり、設計しようとするブロックの面積最小化を優先する場合、何度も配線チャンネルの大きさを変えて試行することになり、レイアウト設計の長ＴＡＴ化の原因の一つとなってしまう。そこでこの問題を解決するものとして、自動配線後の配線チャンネル内の未使用領域を自動で取除くツール、すなわちレイアウトコンパクが必要となる。

コンパクションにおいて、配線の折曲げ（ジョグの挿入）は高圧縮を実現するのに有効な機能である。（図1.1参照）

この配線の折曲げを自動で行なってｘ／ｙどちらか一方向に可能な限り詰めるコンパクション手法［2］も報告されているが、機能ブロックの取扱いに関する記述が十分でなく、またコンパクションによる線長増加に対する考慮も不十分である。

本稿では、ｘ／ｙどちらか一方向に配線の自動折曲げも行なって、グリッドレスで可能な限り詰めるコンパクション手法と、基本素子や機能ブロックや配線等のレイアウト要素が数万個を超えるブロックを対象に本手法を適用した実行結果について述べる。

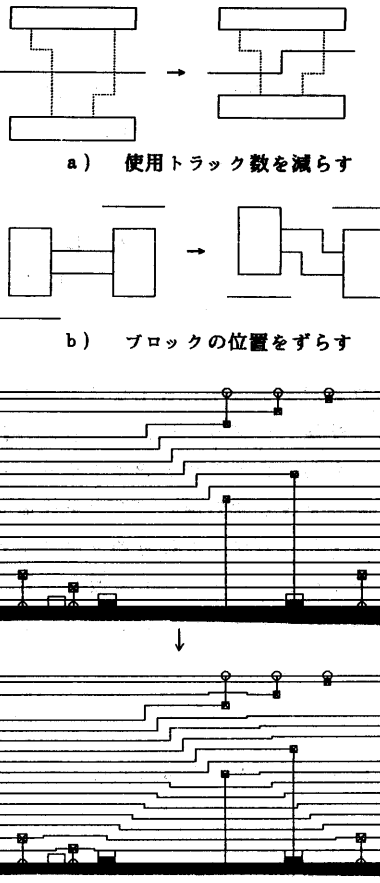


図1.1　配線折曲げの有用性

２．コンパクション手法

2.1　概要

伝統的なコンパクション手法は、コンプレッションリッジ法、制約グラフ法、仮想グリッド法に大別される［3］.本コンパクタは、これらのうち制約グラフ法に属するものである。制約グラフ法とは、レイアウト要素を頂点、レイアウト要素間の隣接関係を枝とし、枝にはレイアウト要素間の最小間隔を重さとして付与してある制約グラフをまず作成し、グラフ上で設計しようとするブロックの一方の外形辺から他のレイアウト要素までの最長路を求め、これをもとに面積最小のレイアウトを得るコンパクション手法である。この方法は利点として

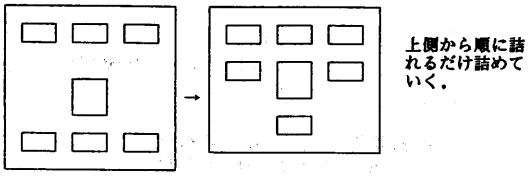
１）一回のコンパクションにより、ｘ／ｙどちらか一方について面積最小のレイアウトが得られる。
２）自由度の高いレイアウトモデルを扱うことが出来る。（例えば、メタル２層以上の多層配線が容易に取扱える。）
３）グリッドレスで処理が出来る。

等が挙げられるが、欠点として

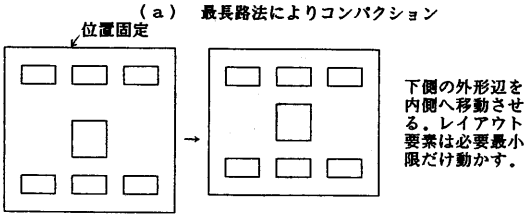
４）コンパクション処理が面積最小だけを目的として行なわれるためコンパクションによってかえって線長が増加する場合もある。

が挙げられる。

本コンパクタも本質的にはこれらと同様の利点、欠点を持つが以下に述べる点が従来の制約グラフ法と異なる。本コンパクタでは、制約グラフを作成するところまでは、基本的には通常の制約グラフ法と同様である。しかし、この制約グラフから面積最小のレイアウトを得る方法が異なる。本手法では、コンパクションは設計しようとするブロックの一方の外形辺の絶対位置を変えないで、他方の外形辺を最小間隔等の制約を満足して、可能な限りブロックの内側に移動させることにより実現されると考える。（図2.1.1参照）このとき、あるレイアウト要素を移動させるのに他のレイアウト要素も移動させなければならないとき、出来るだけ少ないレイアウト要素だけ移動させ、その移動量も最小とする。移動しなければならないレイアウト要素が配線のときは、その配線が移動する原因となったレイアウト要素を移動させるのに、



（ａ）　最長路法によりコンパクション



↑　（ｂ）　本手法によるコンパクション

図2.1.1 最長路法と本手法の違い

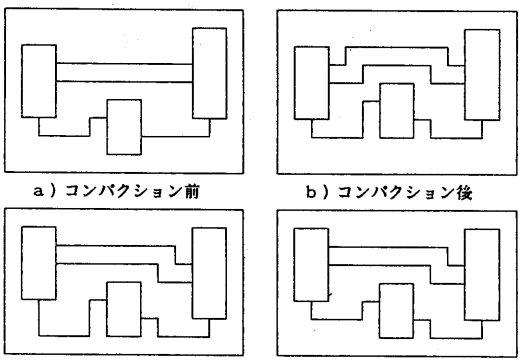


図2.1.2　コンパクションとレイアウトの改善

必要最小限の配線だけ移動すればよい位置で配線を折り曲げる。（図2.1.2参照）こうすることにより、起点となった外形辺をブロックの内側に移動させるのに影響を受けるレイアウト要素の最も少ない移動を行なう。このことは、配線の自動折曲げも行なって可能な限り詰めるコンパクションが実現出来ることを意味する。

しかしながら、配線の折曲げ位置は設計しようとするブロック全体を見て決めているわけではないので、コンパクションには不必要な折曲げを行なっている場合がある。そのためコンパクション後のレイアウトに対して不要な折曲げを取除く処理を行なう。さらにコンパクション処理、不要折曲げ除去処理とも、線長について考慮していないため、線長を低減する処理を最後に行なう。

本章の以降では、制約グラフの定義と作成方法、本コンパクタの基本処理であるレイアウト要素移動手続き、不要な配線折曲げ除去等のコンパクション後のレイアウトの改善のついて述べる。

なお、以降の説明ではコンパクション方向はｙ方向として、レイアウト要素はｙの正方向にのみ移動させるものとする。ｘ方向のコンパクションも同様に処理出来る。また本文中で上側とはｙの正方向をいう。

2.2　制約グラフの定義と作成方法

本節では、制約グラフの定義およびグラフ作成方法について述べる。

2.2.1　制約グラフの定義

制約グフラはレイアウト要素を頂点とし、レイアウト要素が移動する際の制約を枝とするグラフである。以下では、レイアウト要素と頂点をどのように対応させるか、ならびに、本コンパクタの取扱う制約と枝をどのように対応させるかについて述べる。

（ａ）　レイアウト要素と頂点の対応

本コンパクタで取扱うレイアウト要素は、

１．配線（ｙ軸に平行な配線は除く）
２．交点（コンタクトやスルーホール等）
３．下位ブロック
４．配線やコンタクトやスルーホールの禁止領域
５．設計しようとするブロックの外形

である。　本コンパクタではｙ軸に平行な配線は伸縮自在と考えコンパクション後に、その配線の両端にあるコンタクト等のレイアウト要素（交点と呼ぶ）の位置によって、その配線の位置と長さを求めるようにしている。（配線が同一層で折曲っているときや端子に接続しているとき等、配線の端にコンタクトやスルーホールがないときは配線の端の部分を交点と見なす。）よってｙ軸に平行な配線は除く。

基本的にはレイアウト要素と頂点は一対一に対応させるが、次に述べるレイアウト要素には複数の頂点を対応させる。

制約グラフにもレイアウトに対応して層を設定する。各頂点は対応するレイアウト要素の属していた層と同じ層に属させる。ただし、コンタクトとスルーホールについては１個のレイアウト要素に３個の頂点を対応させ，それぞれもとの層と上下の配線層に属させる。そして３個の頂点間に接続制約枝（後述）をはる。

また下位ブロックや禁止領域で外形が短形でないものは短形に分割する。１個の短形以外の外形を持つレイアウト要素に分割された短形数だけの頂点を対応させる。これらの頂点間にも接続制約枝をはる。（図2.2.1参照）

（ｂ）移動制約と枝との対応

本コンパクタではレイアウト要素の移動の際の制約は、

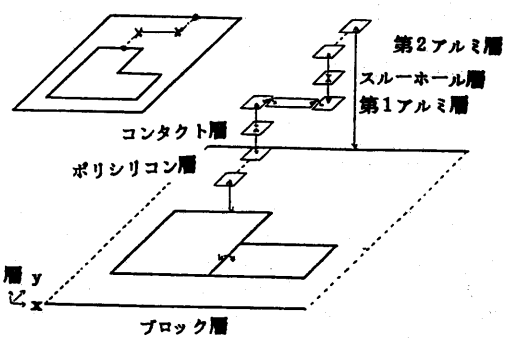


図2.2.1　レイアウトとそれに対応する頂点間の接続関係

a．接続制約　　－　どちらか一方のレイアウト要素が移動したとき他のレイアウト要素も同じ距離移動しなければならない。

b．最小間隔制約　－　２つのレイアウト要素は少なくともこの間隔以上は離れていなければならない。

のみを扱い、頂点間のこれらの制約に対応して枝をはる。(接続制約に対応する枝を接続制約枝，最小間隔制約に対応する枝を最小間隔制約枝と呼ぶ)接続制約枝は、前記（a）ではったもの以外に次に述べるレイアウト要素に対応する頂点間にはられる。

（同一層内）　・接続している配線と交点

（層　　間）　・下位ブロックとそれに設定されている禁止領域
・下位ブロック／設計しようとするブロックの外形辺とそれに接続している配線の交点

枝は有向枝で重さを持つ。接続関係に対応しては図2.2.2に示すように頂点間に向きが互いに逆の２本の重さ０の枝をはる。最小間隔制約関係に対応しては、位置が下側（ｙ座標値の小さい）頂点を始点とし上側（ｙ座標値の大きい）頂点を終点とする向きにはる。重さは、それらに対応するレイアウト要素間にある空いた領域の大きさである。

2.2.2　グラフの作成方法

接続制約枝はグラフの定義よりどの頂点間にはればよいか自明なので、ここでは最小間隔制約枝をはる頂点をどのようにして見付けるかについて述べる。

この処理は、レイアウト要素を移動する際、最小間隔制約に影響をおよぼすｘ座標の範囲を長さする線分をその頂点のレイアウト要素域とし、このレイアウト要素域の相対位置関係を各層毎に平面走査法[4]を用いて調べることにより行なう。

枝はレイアウト要素域のｘ成分の一部あるいは全部を共有し、かつ、間に他のレイアウト要素域がない頂点間にはる。しかしこれだけでは不十分で間に他のレイアウト要素域があっても枝をはらなければならないペアが存在する場合がある。以下ではこのようなペアの見付け方について述べる。図2.2.3のレイアウトにおいてレイアウト要素ｉ、ｊ間の最小間隔をd_{ij}，空いた領域の大きさをl_{ij}とすると

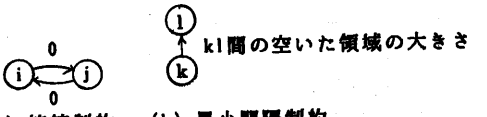
$$d_{13}+l_{13}=d_{12}+l_{12}+d_{23}+l_{23}$$
$$l_{13}=(l_{12}+l_{23})+(d_{12}+d_{23}-d_{13}) \quad (1)$$

(1)式が成立する。今

$$l_{13} \geq l_{12}+l_{23} \quad (2)$$

(2)式が成立するならば、E_1とE_2，E_2とE_3を可能な限り近付けてもE_1とE_3が最小間隔制約を満足する。空いた領域の大きさl_{ij}はレイアウト要素ｉとｊを最小間隔制約を満たして近付ける距離を表しているからである。(2)式が成立するには(1)式より

$$d_{12}+d_{23}-d_{13} \geq 0$$
$$d_{12}+d_{23} \geq d_{13} \quad (3)$$



(a) 接続制約　(b) 最小間隔制約
図2.2.2 枝の向きと重さ

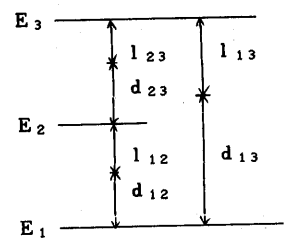


図2.2.3　レイアウト例

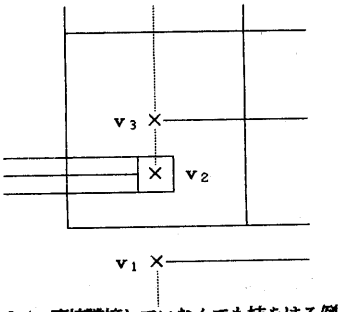


図2.2.4　直接隣接していなくても枝をはる例

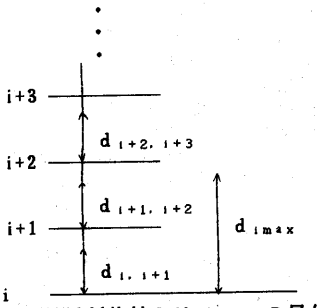


図2.2.5　最小間隔制約枝をはるペアの見付け方

(3)式が成立すればよい。間に他のレイアウト要素がある多くのレイアウト要素のペアでは(3)式の関係が成立するが、図2.2.4の例のような場合(3)式が成立をしないこともある。v_1とv_2が最小間隔制約を満たして近付いてもv_1とv_3が最小間隔制約を満さない場合である。

よって最小間隔制約は、図2.2.5に示すように着目レイアウト要素ｉに設定される最大の最小間隔をd_{imax}としたとき、２番目に隣接するレイアウト要素から順にこれとその間にある最小間隔の累積値（(4)式の左辺）を比較していき、(4)式が成立する場合、そのレイアウト要素域と着目レイ

アウト要素域に対応する頂点間にはる。この処理は(4)式が成立しなくなるまで続ける。

$$d_{i,i+1} + \sum_{k=i+2} d_{k-1,k} < dimax \qquad (4)$$

2.3 レイアウト要素移動手続き

本節では、本コンパクタの基本的な手続きであるレイアウト要素移動手続きについて述べる。本手続きの機能は、

a. 移動させたいレイアウト要素(起点レイアウト要素)

b. 移動させたい距離と方向(要求移動量)

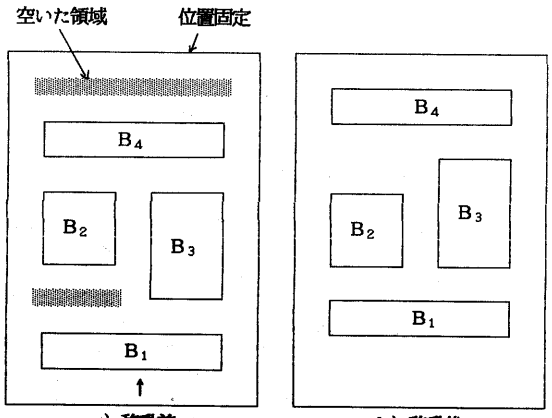
を入力とし、要求移動量を限度として可能なだけ起点レイアウト要素を移動させる。ここで可能なだけとしているのは移動が禁止されているレイアウト要素（位置固定レイアウト要素）が入力レイアウト中に存在し要求移動量を満たさない場合があるからである。

コンパクションは、第2.1節で述べたように設計しようとするブロックの上側の外形辺を位置固定とし、下側の外形辺を起点レイアウト要素、要求移動量をブロックの上側へ無限大として本手続きを適用することにより実現出来る。コンパクション後のレイアウトの改善（不要配線折曲げ除去や線長低減）は本手続きをレイアウトの一部分に適用することにより実現する。（図2.1.2参照）レイアウトの改善にも本手続を用いるのは、下位ブロックの移動も含む改善が容易に取扱えるからである。

以降では、まず本手続を実現するための基本的な考え方を簡単な例を用いて述べた後、手続きの詳細について述べ、最後に配線の折曲げも行なえるようにこれを拡張する。

2.3.1 基本的な考え方

図2.3.1のレイアウトを例にレイアウト要素の移動をどのようにして行なうかについて述べる。ここでは、問題を簡単にするためにレイアウト要素としては下位ブロックのみを考え、これらは1単位距離だけ離れていれば、最小間隔制約を満足するものとする。また、移動方向はyの正方向、要求移動量は2、起点レイアウト要素をB_1とする。



a) 移動前　　b) 移動後

図2.3.1レイアウト要素の移動

B_1を2だけ移動させるにはB_1の上側に2だけの空いた領域があればよい。しかしこの例のようにない場合はさらに上側にあるB_2, B_3も移動させなければならない.このときB_2の要求移動量はB_1との間の空いた領域が1だけあるので残り1.B_3の場合は0しかないので2のままとなる。B_2, B_3とも上側に空いた領域がないのでさらにB_4も移動させなければならない。B_4にはB_2, B_3と2つの移動要求が伝播してきたが、このときは当然要求移動量の大きい方を優先しなければならない.この場合ではB_2からの要求移動量が1、B_3からの要求移動量が2であるから B_4の要求移動量は2となる.B_4の上側には1の空いた領域しかないので、移動要求は移動が禁止されている設計しようとするブロックの上側のブロック外形辺まで伝播してくる。空いた領域を見付ける探索はここで終了する。

各ブロックがどれだけ移動出来るかはそれらの上側にどれだけの空いた領域を見付けることが出来たかによって定まる。前述のような空いた領域を見付ける探索時に各ブロックに起点のブロックの要求移動量からそのブロックの要求移動量を引いたもの（起点のブロックからそのブロックの間にある空いた領域の大きさに相当）をコストとして付与しておく。こうすると各ブロックの上側にある空いた領域の大きさは移動が禁止されている上側ブロック外形辺の重さ Cmaxから各ブロックiのコスト c(i)を引くことにより得られる。よって各ブロックiの移動後のy座標値Y(i)はもとのy座標をy(i)とすると

$$Y(i) \leftarrow y(i) + Cmax - c(i) \qquad (5)$$

(5)式で与えられる。この例では、空いた領域を見付ける探索が、移動を禁止されているレイアウト要素に至ったので終了したが、全てのレイアウト要素の移動要求が満されて探索が終了したときはCmax は起点レイアウト要素の要求移動量とすればよい。

以上のようにレイアウト要素を移動させるためには、空いた領域を見付けることが必要となるが、本コンパクタでは、この空いた領域を見付ける探索を制約グラフ上で行なう。

2.3.2 レイアウト要素移動手続きのアルゴリズム

第2.3.1節で述べた空いた領域を見付ける探索は、第2.2節で述べた制約グラフ上で起点レイアウト要素に対応する頂点から順に最短路を見付けることにより実現される。グラフの枝には始終点に対応するレイアウト要素間にある空いた領域の大きさを重さとして付与している。全てのレイアウト要素は最小間隔制約を満足しているから重さは非負である。（もし仮に負のものがあったときは0として一すなわちレイアウト要素間の間隔を拡げることはしないで一処理を進める。）枝の重さが非負であるグラフ上で最短路を求めるアルゴリズムとしてはダイクストラの方法がある。本コンパクタでは、レイアウト要素への移動要求が全て満たされたときあるいは移動要求が位置固定レイアウト要素に至ったとき探索を終了するようにダイクスト法を一部修正して空いた領域を見付ける探索に用いている.図2.3.1にこのアルゴリズムを示す.

ここでs.valは起点レイアウト要素に対応する頂

```
PROCEDURE LAYOUT MOVE
BEGIN
 T←φ:Cmax←val:
 c(s)←0:put s in WFL
 WHILE WFL is not empty DO
  BEGIN
   get x from WFL that has min. cost
   IF x is fixed position vertex THEN
    BEGIN
     Cmax←c(x):GOTO L1
    END
   IF x ∉T THEN
    BEGIN
     T←T∪{x}
     FOR each p that(x,p) ∈E DO
16     IF p ∉T and
        val>c(x)+w(x,p) THEN
        BEGIN
         c(p)←c(x)+w(x,p);
         put p in WFL
21       END
    END
  END
L1: FOR each t∈T DO y(t)←y(t)+Cmax-c(t)
 END
```

図2.3.2 レイアウト要素移動アルゴリズム

点、要求移動量であり、E,Tは制約グラフの枝、探索木であり、wは枝の重さである。またWFLはコストをキーとするプライオリティーキューであり取出される頂点はキュー中の最小コストを持つ頂点である.

2.3.3 配線折曲げのための拡張

次に配線の折曲げ（ジョグの挿入）処理について述べる.第2.1節で述べたように移動しなければならないレイアウト要素が配線のとき、その配線が移動するもととなったレイアウト要素を移動させるのに必要最小限の配線だけ移動すればよい位置にジョグを挿入する。ジョグの挿入によって分割された配線の各部分（部分配線）は以降、それぞれ1個の頂点として処理を続ける.図2.3.3の例でw_2', w_2''が部分配線である。

配線の分割によって今まで間に他のレイアウト要素があり隣接していなかったレイアウト要素と部分配線が隣接する可能がある。そのためジョグ挿入位置の上下にあるレイアウト要素と部分配線の間に枝をはる。図2.3.3の例では、B_0とw_2', w_2''とw_1の間に枝をはる。本アルゴリズムでは、探索木に次の葉として加える頂点から出ている枝を全て調べてWFLへ格納する。よって新たにはった枝の始点が既に探索木に属していれば終点をpとして.枝を調べVFLへ格納する処理(図2.3.2のアルゴリズム中の16～21行の処理）を行なう。

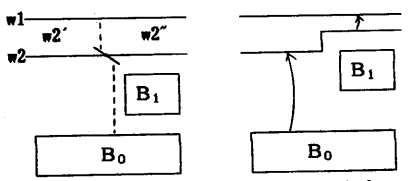


図2.3.3 配線折曲げ時の枝のはり方

2.4 コンパクション後のレイアウト改善

2.4.1 不要な配線折曲げの除去

第2.3節で述べた方法でコンパクションした場合、配線の折曲げ位置を局所的なレイアウトだけ見て決めているので、レイアウトの圧縮に貢献しない不要な配線折曲げを生じる.（図2.1.2参照）以下ではこの不要な配線折曲げを取除く方法について述べる。

不要な配線折曲げを取除くのにもレイアウト移動手続を適用して行なう。レイアウト要素移動手続を用いることにより、下位ブロックの移動も伴なう不要な配線折曲げの除去が容易に行なえる。

不要な配線折曲げの除去は、まず折曲りを持つ配線一本毎に着目していきこれがまっすぐになるように各部分配線を引上げることにより行なう。

折曲りを持つ配線の着目順序は、コンパクション前のｙ座標値の大きい順とする。

この方が全くランダムな順序で行なうよりもレイアウト要素移動処理の扱う問題の規模が小さいことが期待出来るからである。

一本の折曲りを持つ配線に着目した後、この配線の各部分配線をどの順序で起点レイアウト要素するかについて述べる。

図2.4.1に示すように部分配線の処理順によって取除かれる折曲りと処理後の配線の形状が異なる。一般に部分配線のｙ座標が極大から極小の順に行なえば極小から極大の場合に比べてより上側に配線を引上げるとが期待出来るが、減らせる折曲りの個数が少なくなる場合が多い。本コンパクタでは、部分配線がより上側に上がることによって空いた領域は下側の配線の不要折曲げ除去に利用出来る点に着目し、極大から極小の順で処理している。

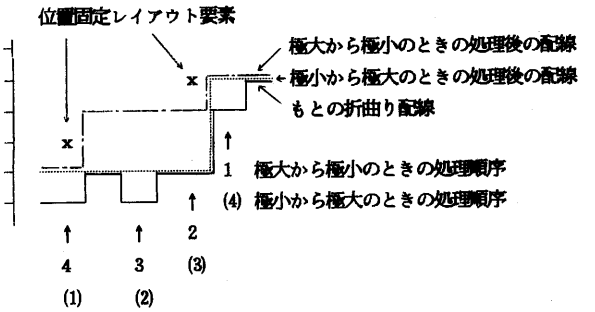
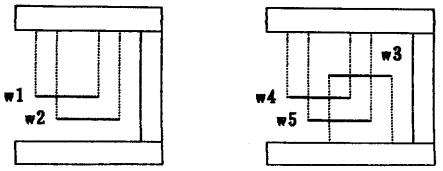


図2.4.1 部分配線の処理順序

2.4.2 線長の低減

コンパクション処理、不要な配線折曲げ除去処理ともに処理によってｙ軸に平行な配線の線長が増加することについて考慮していない（図2.1.2参照）。よって、これらの処理が終った後、ｙ軸に平行な配線の線長を短かくする処理を行なう。

本コンパクション手法では配線が相対的に下側に詰まる傾向にあるため、上側に配線を引上げることによりｙ軸に平行な配線が短かくなるか調べ、短かくなるときこれを移動させることにより線長低減を行なう。



a)ローカルな判定でよいとき　b)グローバルな判定がいるとき

図2.4.2 線長低減の判定

線長低減もコンパクション前のｙ座標値の大きい配線順に着目してこれにレイアウト要素移動手続きを適用して行なう。ただし、ここではレイアウト要素移動手続きを一部修正して用いる。線長を短かくするのに上側から順に配線を調べて両側の交点の上側にｙ軸に平行な配線があるときこの配線を上側のレイアウト要素まで移動させるとする。図2.4.2 a)の側の場合はこれでよいが、同図b)の場合、配線はw_3，w_4，w_5とも上側に移動した方が線長が短かくなることもかかわらず移動はしない。本コンパクタではレイアウト要素移動手続きを修正し、同時に同じ量だけ移動するレイアウト要素群単位で線長がその移動によって短かくなるかを判定し、短かくなるときだけ移動させるようにして線長低減処理に用いている。こうすることにより図2.4.2 b)のw_3，w_4が起点レイアウト要素のときは移動は起らないが、w_5が起点レイアウト要素のときはw_3，w_4，w_5が移動する。　なお、本コンパクタでは層毎に線長低減優先度が設定出来る。ポリシリコン層の優先度を他の層より大きく設定して，ポリシリコン層の配線を他の層の配線より優先して短かくするようにしている。

3．　実行結果

ポリセルタイプのブロックB_1～B_4をコンパクションしたときの圧縮率と処理時間を表 3.1に示す。グリッドレスのコンパクションはグリット間隔をレイアウト要素移動の単位距離とするコンパクションより５％程度より圧縮出来た。図1.１ c)に示した細かい配線折曲げの効果と思われる。なお処理時間はグラフより最終レイアウトを得るまでについて示している。使用計算機の処理速度は15ＭＩＰＳである。

図 3.1にコンパクション前と後のレイアウト例を示す。なお紙面の都合でブロックの一部のみを示す。このブロックのゲート数は５ＫＧ、配線数は14.530本であり所用処理時間は139秒であった。

	データ規模		圧縮率		処理時間	
ブロック	ゲート数	配線数	grid on	gridless	grid on	gridless
B_1	0.4k	1595	0.0%	4.3%	7秒	14秒
B_2	0.8k	2512	3.6%	10.2%	12秒	28秒
B_3	0.9k	2595	0.5%	6.8%	8秒	24秒
B_4	1.2k	2346	0.0%	5.4%	8秒	31秒

表3.1　コンパクション結果

4．　まとめ

本稿では、ｘ／ｙどちらか一方向に配線の自動折曲げも行なって可能な限り詰めるコンパクション手法とその実行結果について述べた。本手法を、レイアウト要素数が数万を超えるレイアウトに適用しても十分実用的な処理時間で処理出来ること、また，配線の自動折曲げがグリッドレスコンパクションの効果を５％程度高めることが確認出来た。

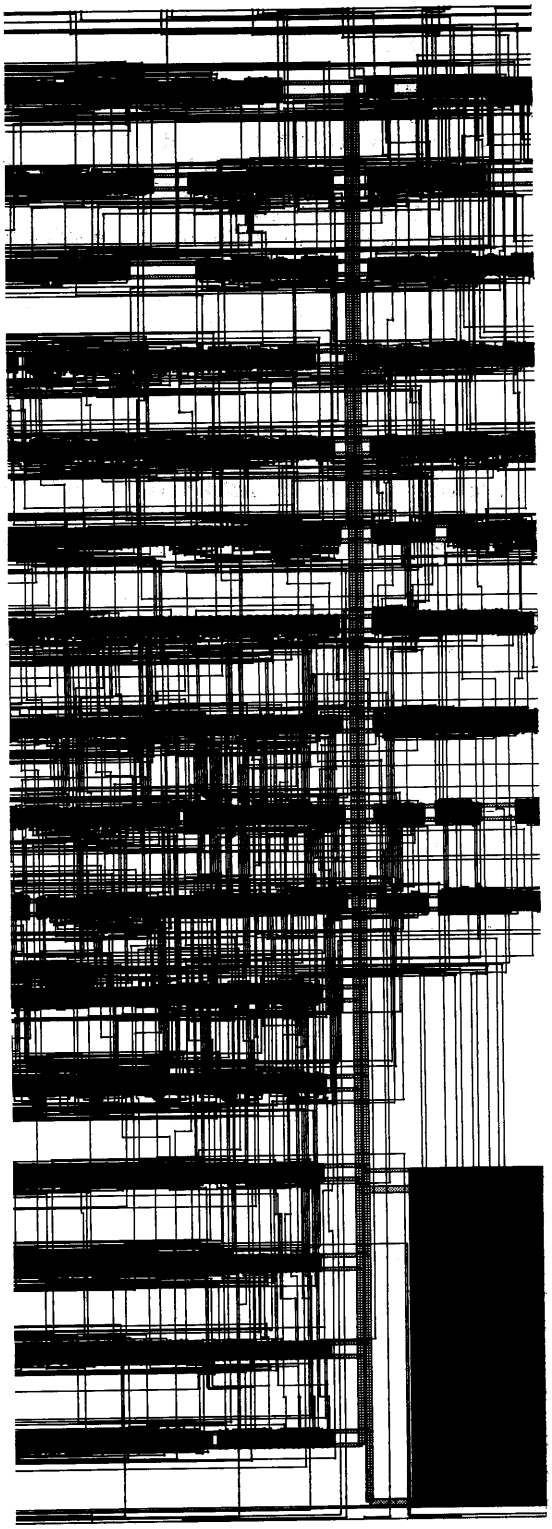
【参考文献】

[1]荘司、他:"VLSIレイアウトシステム(VILLA)の構成."情報処理学会第29回全国大会論文集. pp. 1671-1672. 1984

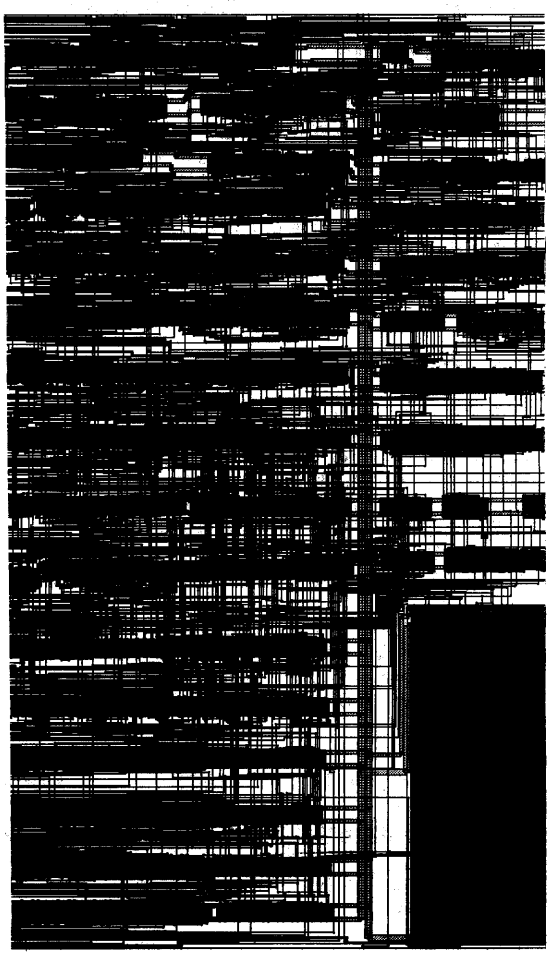
[2]鎌田、他:"多層レイアウトパタンに対する高機能コンパクション手法."信学技報，CAS86-55. pp. 39-45. 1986

[3]Cho. Y. E. :"A Subjective Review of Compaction." Proc. of 22nd DA Conf. , pp. 396-404. 1985

[4]Baird. H. S. , et al:"An Artwork Design Verification System."Proc. of 14th DA Conf. , pp. 414-420. 1975



a)　コンパクション前



b)　コンパクション後

図3.1　コンパクション例