設計自動化 48－5
（1989. 7. 11）

知識処理応用アナログＬＳＩレイアウト手法

加藤 直樹　茂垣 真人

日立製作所

アナログＬＳＩのレイアウトでは、回路特性を保障するために、多くの制約を守る必要があり、そのアルゴリズム化が難しい。本稿では、この問題の解決策として、レイアウト改善に知識処理を応用する一方式を提案する。本方式は、制約違反を修正するルールと、面積を縮小するルールを知識ベース内に持たせ、レイアウトとこれらのルールの照合、実行を繰り返すことにより改善を行なうものである。効率的な照合、効果的なルールの選択を可能とするため、新たなルールの表現法（動的集合）と推論方式（予測推論）を提案した。

Common-Lisp を用いて大型計算機上にプロトタイプシステムを作成した。７１素子からなるブロックに適用した結果、レイアウト制約を遵守し、かつ初期レイアウトに対して７％の面積縮小ができた。

AN AUTOMATIC LAYOUT TECHNIQUE FOR ANALOG LSI's USING THE KNOWLEDGE BASED APPLOACH

Naoki Kato and Masato Mogaki

Hitachi, Ltd.

1-280 Higashi Koigakubo, Kokubunji, Tokyo 185, Japan

It is very difficult to devise an efficient algolithm for an analog LSI layout because circuit performances as well as the layout area must be optimized. In this paper, a new automatic layout technique is proposed and experimentally applied to an actual analog LSI layout problem. Its characteristic is the application of knowledge-based technique to the layout inprovement process. This technique improves the layout by repeated matching and application of such rules that eliminate layout-constraint violations and minimize the layout area. A new rule-representation and an inference mechanism, called a "dynamic set" and a "predictive inference", respectively, are also suggested to make the rule-matching and rule-selection effective.

A prototype is developed by using Common Lisp on a large mainflame computer. This is applied to the desegn of block consisting of 71 cells and it has improved the block area 7% comparing with the initial one.

１．はじめに

デジタルＬＳＩの自動レイアウトでは、面積最小化を目的に、多くのアルゴリズムが考案され、人手並の面積を実現する実用システムが開発されてきた[1]。ところが、アナログＬＳＩについては、デジタルＬＳＩに比べ、レイアウトの自動化が遅れている。近年、幾つかのシステム[2]~[5]が開発されているが、その適用範囲に制限があり、多くは人手に頼っているのが現状である。アナログＬＳＩのレイアウトが困難な最大の理由は、レイアウトの良し悪しが回路特性に大きく影響することによる。つまり、単に結線関係を守ったレイアウトを行なえば、所望する回路特性が得られるわけではないということである。そのため、レイアウト設計者は、回路の種類と特徴に応じたレイアウトの幾何学的条件（以下、レイアウト制約と呼ぶ）を設定し、その制約に従いレイアウト設計を行なっている。

アナログＬＳＩの自動レイアウトを実現するためには、このレイアウト制約のもとで、面積の最小化を行なわなくてはならない。

我々はこの問題に対して、制約を「知識」として計算機に組み込み、この「知識」をシステムが必要に応じて選択、実行する知識処理の応用が有効であると考え、アナログＬＳＩのための知識処理応用レイアウト手法について検討を行った。本稿では、その基本概念、知識処理方式として提案する動的集合と予測推論機構、およびその評価結果について述べる。

２．基本概念

アナログＬＳＩの自動レイアウトを実現するために我々が提案する方式では、レイアウト設計を生成過程と改善過程に分けている。これは生成過程で全ての制約を満たしたレイアウトを得るのが、以下の理由により困難であるためである。

（１）レイアウト生成のアルゴリズムで用いるモデルでは取り扱いにくい制約が存在する。

（２）制約を組み込むことにより、レイアウト面積が増大する傾向がある。

そのため、改善過程では、レイアウトパターンから制約違反箇所や面積的に損な部分レイアウトを抽出し、抽出した箇所に対して改善を施す処理を繰り返す。ここでは、予め改善を行なう順序を決定しておくことは難しいので、手続き的なアプローチは適さない。むしろ、以下の様な知識処理の利用が適している。

（１）改善を要するパターンとその改善手順を組にして知識として用意する。

（２）知識とレイアウトの照合を行ない、問題のある部分レイアウトを抽出する。

（３）抽出された複数の部分レイアウトの中から一つを選択し、改善を実行する。

この方法によれば、レイアウト制約は、知識として知識ベースに蓄えておき、制約の変更、追加には知識ベース内の知識の変更、追加で対応することができる。

また、一般的に知識処理は手続き処理よりも処理速度が遅いという問題があるが、改善処理では解の探索範囲が制限でき、知識処理応用の実現性が高いと考えられる。

３．改善処理方式

３．１　基本構成

知識処理により、レイアウト改善を行なうために必要な基本構成を図３．１に示す。

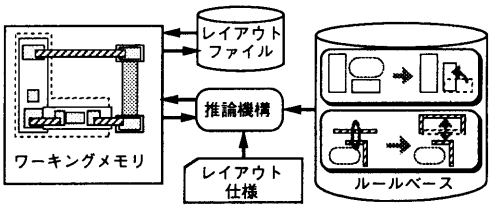


図 3.1　基本構成

入力となるのは、レイアウト仕様と生成過程で作成されたレイアウトファイルである。レイアウト仕様とは、回路対応に与えられるものであり、その回路で守るべきレイアウト制約に関する情報である。たとえば、「トランジスタＱ１とＱ２はペア素子として、隣接配置しなくてはならない。」とか「信号線１と２は交差して配線してはいけない。」等の情報を持っている。

基本的な構成要素は以下の３つである。

（１）ワーキングメモリ：入力されたレイアウトデータを内部で取り扱うデータ表現形式に変換して格納する領域。

（２）ルールベース：改善知識を格納した領域。

（３）推論機構：改善ルールとワーキングメモリ内のデータとの照合を行ない、マッチするルールを実行する機構。推論機構は改善すべき箇所が存在しなくなるまで、この照合・実行のサイクルを繰り返す。

３．２　レイアウトデータの表現

レイアウトデータはオブジェクト指向の考えを取り入れ、Lispのデータ型である「構造(structure)」を用いて表現した。データはレイアウト対象であるトランジスタ、抵抗等の素子、素子内の端子、外部端子、同電位端子間を結ぶ配線、スルーホール等である。

①オブジェクトの構造…各オブジェクトは、「素子」や「配線」等のクラスに分類される。同じクラスに属するオブジェクトは、同じ種類のスロットを持ち、スロットにはそれぞれのオブジェクトにより異なった属性値を書き込む。オブジェクトによる表現例を図３．２に示す。図３．２の（ａ）は、トランジスタｑ513を表わしているオブジェクトである。q513は、「素子」というクラスに属し、素子タイプは npn-トランジスタであり、配置されている座標値、寸法、端子位置等を各スロットに書き込んだ属性値により表わしている。（ｂ）は、配線を表現するオブジェクトの一例である。

②メソッド…各クラスは、メソッドと呼ばれる操作手続きを持つ。メソッドの例としては、「生成」、「削除」、「移動」、「表示」等の操作手続きがあり、オブジェクトがそれらの命令を受けると、クラスのメソッドの定義に従い処理が実行される。これを具体例により説明する。「Ａ、Ｂ間を配線でつなぐ。障害物が存在すれば、障害物を移動する。」という処理を行なう場合を考える。Ａ，Ｂ間の障害物が、配線、素子、スルーホール、外部端子、のいずれかによりそれぞれ移動処理が異なる。しかし、処理上は障害物に「移動」の命令を送れば良い。「移動」の命令を受けたオブジェクトは自分クラスの「移動」メソッドの定義に従い処理を行なう。もし、障害物が素子である場合、素子クラスの「移動」メソッドを実行する。素子クラスの移動メソッドでは、素子が有する端子の移動も行なうために、自分の端子に「移動」の命令を送る。端子オブジェクトは、端子クラスの移動メソッドに従い移動を行なう。

③クラスの階層構造…クラス間には階層関係があり、下位のクラスは上位のクラスの性質を継承する。上位のクラスはより抽象度の高い情報を持ち、下位のクラスはより具体的な情報を持つ。具体的には、下位のクラスは、上位のクラスの持つスロットを全て継承し、それに加えてより具体的な概念を表現するために上位のクラスが持たないスロットも持つ。クラス間の階層関係例を図３．３に示す。この例では、矩形領域クラスの下位に配線対象物クラスがあり、配線対象物クラスの下に配線クラスとスルーホールクラスがある。矩形領域クラスは４つの座標値、xmin,xmax,ymin,ymax のスロットを持つ。配線対象物クラスは、矩形領域であるのに加えて、配線層、ネットのスロットを持つ。さらに、配線クラスは、幅、方向のスロットが加わる。また、下位のクラスは上位のクラスのメソッドも継承する。この階層関係により操作手続きの定義を重複して記述することがない。例えば、２つの矩形領域が交わりを持つかを調べる場合には、その矩形領域が素子であるか配線であるかを考える必要はない。矩形の交わりを調べるメソッドは上位のクラスである矩形領域クラスで定義し、下位クラスである素子、配線、スルーホールはそのメソッドをそのまま継承すれば良い。

クラス	素子
super class	配置対象物
name	q513
x-point	25
y-point	14
type	npnトランジスタ
terminal	(526-c,526-b,526-e)
x-size	25
y-size	40

クラス	配線
super class	配線対象物
name	wire12
direction	x
start	(21,45)
end	(86,45)
width	6
net	net13
layer	Al1

（ａ）素子クラスのオブジェクト　（ｂ）配線クラスのオブジェクト

図 3.2 オブジェクトによるデータ表現例

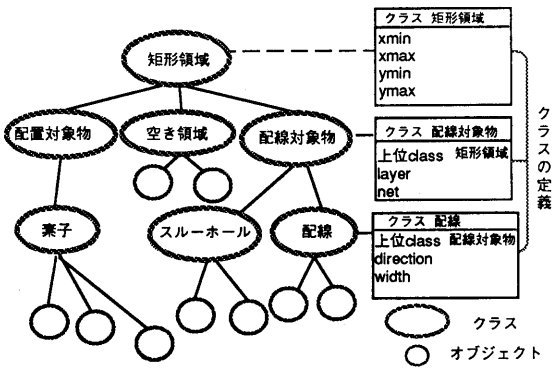


図 3.3　クラス間の階層関係

３．３　ルールとその記述言語

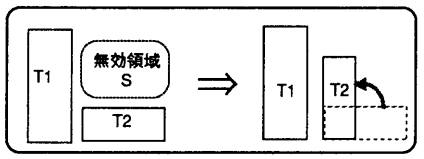
（１）ルールの種類　次に示す２種類ルールがある。

①面積縮小ルール…面積縮小ルールは、ＩＦ部に面積的に損なレイアウト状態を示す素子や配線の組合せパターンを記述し、ＴＨＥＮ部に、そのパターンに対する改善方法を記述する形式をとる。図３．４（ａ）に面積縮小ルールの一例を示す。図のルールは、２つ

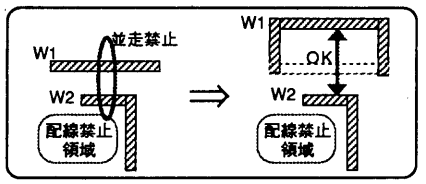
のトランジスタＴ１，Ｔ２と無効領域Ｓからなる部分レイアウトに対する改善ルールである。トランジスタＴ２を９０度回転することにより、無効領域Ｓをなくし面積を縮小する。

②制約遵守ルール…制約遵守ルールは、ＩＦ部にレイアウト制約に違反する素子や配線の組合せパターンを記述し、ＴＨＥＮ部は改善方法を記述する。図３．４（ｂ）に制約遵守ルールの一例を示す。制約の一つに信号間の干渉を防ぐために、ある配線間には基準値以上の距離を並んで配線されることを禁止する場合がある。図のルールはその制約に違反して配線されている場合に、片方の配線を迂回させて配線の並走をなくすことを記述したルールである。

ルールベースには上の２種類のルールが複数格納されている。



(a) 面積縮小ルール



(b) 制約遵守ルール

図 3.4　改善ルール例

（２）動的集合

動的集合はルールの照合を効率化するために導入したデータである。

上述のように、データはオブジェクトにより表現した。一方、ルールはＩＦ－ＴＨＥＮ型式の表現である。ルール条件部とオブジェクトの照合を行ない推論を進める。あるルールにマッチするオブジェクトを見付けるのに、およそ関係がないクラスのオブジェクトとの照合を行なっていたのでは、処理時間を無駄に費やしてしまう。また、複数のルールに共通した条件の組み合わせが存在する場合、それぞれのルール毎に照合を繰り返すのも処理時間の無駄遣いである。そこで、新たに考案した動的集合の概念を導入し、推論の効率化を狙った。

動的集合は、ある条件を満たすオブジェクトの集合であり、ルールで照合されるレイアウト条件を予めオブジェクトの中から抽出しておき、集合として保持するものである。

動的集合は、ベース集合、ベース集合の部分集合、およびそれらの交わりと結びから成る。ベース集合とはクラスに属するオブジェクト全体の集合である。例えば、素子クラスのベース集合は、素子を表すオブジェクト全てを含む集合である。また、npn-トランジスタの集合は素子クラスのベース集合の部分集合として定義でき、素子クラスに属するオブジェクトの内、素子タイプスロットの属性値が npn-トランジスタであるものの集合である。これも、動的集合である。また、「スルーホールとそのスルーホールによって接続するＡ１１層配線とＡ１２層配線」という動的集合も定義できる。このような集合を予め定義しておくと、複数のルール条件部にこの条件が記述される場合には、それらのルールで共通にこの動的集合を参照すれば良い。動的集合という名前は、集合の要素であるプリミティブなオブジェクトが「削除」、「移動」、「生成」等のメソッドで変化した場合、定義に従い動的に集合が作り替えられる機能を持つことに由来を持つ。動的集合によりルール条件部の共通条件のくくりだしを行なうことで、ルールの記述量の削減と条件の抽象化が可能となる。

（３）ルール記述言語

ルールは動的集合の考えに基づくルール記述言語を設計し、Lisp上にインプリメントした。

図３．５にルール記述言語のシンタックスを示す。以下に、その記述例を示し、説明する。

```
①ベース集合
    @wireseg                ;配線クラスのベース集合
②動的集合の定義
    <例:A12層配線の集合>
   (Defset  A12-wireseg     ; 集合名はA12-wireseqとする。
        W|W<-@wireseg       ; Wは集合@wiresegの要素である。
        W.layer = 'A12)     ; Wのlayerスロットの値はA12である。
③ルールの定義
    <例:A12層配線をA11層に変換するルール>
(Defrule  A12-to-A11  ; ルール名A12-to-A11。
    W<-@A12-wireseg   ; Wは@A12-wiresegの要素である。
    % W@underrun      ; Wと交わるA11層配線が存在しない。   ｜ＩＦ部
    % W@underterm     ; Wの下に端子が存在しない。
   /% W@connect-th    ; Wはスルーホールをもつ
       -->            ;       ならば
   (change-layer W)) ; Wに層変換の命令を送る。            ｜ＴＨＥＮ部
```

＜説明＞

上の例ではＡ１２層配線の動的集合を用いて層変換のルールを定義している。動的集合は集合名および条件により定義する。例では変数Ｗを配線クラスのベー

ス集合の要素とし、Wのlayerスロットの値がAl2という条件が書かれている。ルールは、既に定義した動的集合を用いて条件を記述する。例では変数WをAl2配線の集合の要素とし、Wについての動的集合、W@underrun、W@underterm がそれぞれ空集合であるという条件とW@connect-thが空集合でないという条件を満足すれば、change-layerの命令がWに送られてWの配線層が変換されることを表している。

```
(1) ルールの定義
〈ルール〉    ::= (DEFRULE 〈ルール名〉(〈起動条件〉 …)
                    〈条件〉 …
                    −>
                    〈操作〉 …)
〈操作〉      ::= 〈変更〉 | 〈追加〉 | 〈変更〉| 〈代入〉 | 〈操作呼出〉
〈追加〉      ::= NEW 〈ベース集合名〉
〈削除〉      ::= DEL 〈要素変数〉
〈変更〉      ::= 〈要素変数〉.〈スロット〉 ::= 〈式〉
〈代入〉      ::= 〈変数〉 := 〈式〉
〈操作呼出〉  ::= 〈操作名〉 (〈式〉 …)
(2) 動的集合の定義
〈動的集合〉  ::= (DEFSET 〈集合名〉(〈仮パラメータ〉 …)
                    〈条件〉…
〈集合〉      ::= 〈有名集合〉 | 〈無名集合〉
〈有名集合〉  ::= @〈集合名〉
                 | @〈集合名〉 (〈実パラメータ〉 …)
〈無名集合〉  ::= (〈要素〉 ! 〈条件〉 …)              無順序集合
                 | [〈要素〉 ! 〈順序〉/〈条件〉 …]     順序集合
                 | 〈集合〉 !! 〈集合〉                和集合
                 | 〈集合〉 && 〈集合〉                積集合
〈要素〉      ::= 〈変数〉
                 | [〈変数〉 …]                       順序対
                 | (〈変数〉 …)                       無順序対
〈順序〉      ::= 〈比較述語〉                         述語により定まる順序
                 | 〈比較述語〉 :KEY 〈位置指定子〉   (SORT の2番め以降の引き数)
〈条件〉      ::= 〈要素〉 <− 〈集合〉                  要素が集合の元である
                 | /% 〈集合〉                        集合が空でない
                 | % 〈集合〉                         集合が空である
                 | 〈要素〉 /== 〈集合〉               要素が集合とマッチしない
                 | 〈要素〉 == 〈集合〉                要素が集合とマッチする
                 | 〈述語〉 ( 〈式〉 …)                述語が真である
                 | 〈式〉 〈比較子〉 〈式〉             比較が真である
                 | not 〈条件〉                       条件が成り立たない
                 | 〈条件〉 & 〈条件〉                 条件が共に成り立つ
                 | 〈条件〉 ! 〈条件〉                 どちらかが成り立つ
                 | 〈変数〉 := 〈式〉                  代入 (常に成り立つ)
〈比較子〉    ::= = | /= | <= | < | >= | >
〈式〉        ::= 〈項〉| (式) 〈演算子〉 〈式〉
〈演算子〉    ::= + | - | * | /                       演算子の優先順位は
                                                     通常の規則に従う
〈項〉        ::= 〈変数〉
                 | 〈変数〉.〈スロット〉
                 | (〈関数〉 〈式〉 …)                LISP関数呼出し
```

ここで用いた記法は、拡張ＢＮＦ(Bakus Normal Form)で、[と]で囲まれた要素は、あっても無くても良いもの、|で区切られた要素は、そのいずれかを選択することを示している。

図３．５　ルール記述言語シンタックス

３．４　予測推論機構

予測推論機構は効果的な改善を実現するために考案した推論機構である。

ルールを用いて改善を行なう場合、単純にマッチしたルールを実行するのでは効果的な改善を行なうことはできない。また、場合によってはルール実行前の状態よりも悪くなることが生じる。例えば、面積縮小ルールを適用することにより、今まで制約を守ってレイアウトされていた箇所が、制約違反を侵してしまうこともある。これについて図３．６の例を用いて説明する。（ａ）は初期レイアウト状態を示す。この状態に

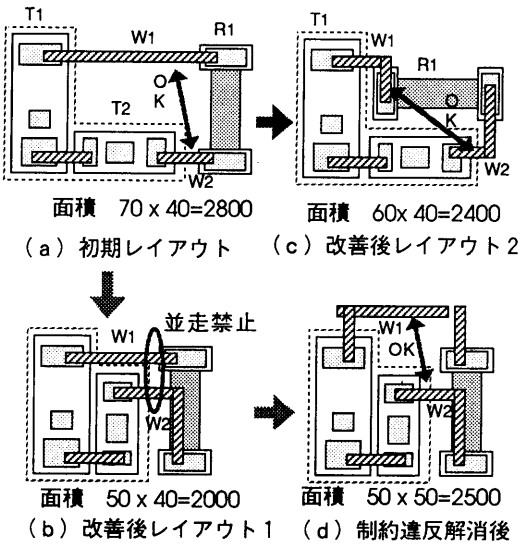


図 3.6　ルールの選択

対して（ｂ）の状態への改善と（ｃ）の状態への改善のどちらかを選択することができる。（ｂ）はＴ２を９０度回転し無効領域を削減するルールの適用結果であり、（ｃ）はＲ１を９０度回転することでやはり無効領域を削減するルールの適用結果である。改善後の面積は改善前を１とすると、（ｂ），（ｃ）ではそれぞれ０．７１、０．８６となる。面積の縮小率のみを考えると（ｂ）の改善の方が効果的であり、（ｂ）への改善を行なうルールが選択されることが望ましい。ところが、（ｂ）の図で配線Ｗ１とＷ２に並走禁止の制約がある場合は、制約を守るためにさらに（ｄ）の状態への改善が行なわれる。（ｄ）は（ａ）に対する面積比０．８９となり、結果的には（ｃ）の方が良い改善であったことになる。

このような場合に（ｃ）の方を選択するためには、一つのルールを適用した結果、どのようなレイアウト状態になるのかを知る必要がある。しかし、各ルールは局所的な部分レイアウトに対しての改善を行なうものであり、ルール適用後の周囲への影響を考慮していないため、予め適用結果を知ることはできない。また、ルール間の優先度は決定的なものでなく、レイアウト状態と組み合わせて始めて比較できるものであるために、予めルールにプライオリティを付けることはできない。そのため、状況に応じて最適なルールを選択するには、推論機構にルール適用後の改善の効果を評価する機能を組み込む必要がある。しかし、市販の汎用の知識処理言語の推論機構では、予め決定された優先

順位によりルールを選択することしかできない。そこで、上のような機能を持つ推論機構として、予測推論機構を考案した。

図３．７に推論機構の概要を示す。

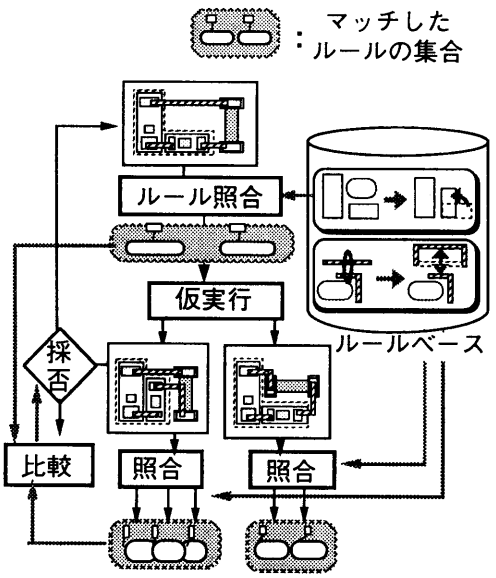


図 3.7　予測推論機構

推論は以下のステップで行なう。

Step 1. レイアウトパターンとルールベースを照合して、マッチするルールの集合を作成する。

Step 2. マッチしたルールの１つについて仮の実行を行なう。ルール適用前のデータは保存しておく。

Step 3. 仮実行結果のレイアウトパターンとルールベースの照合を行ない、マッチするルールの集合を作成する。

Step 4. 改善前のマッチするルールの集合とルール仮実行後のマッチするルールの集合を比較する。

Step 5. 改善前よりも、マッチするルールが増加する場合や新しく重大な制約違反が生じる場合は、そのルールの採用をやめ、次の候補に対してstep２から繰り返す。逆に、マッチするルールが減少するならば、そのルールを採用することにして実行し、step1から繰り返す。

予測推論機構ではルールを仮に実行し、ルール適用後のレイアウト状態を評価して採否の判定を行なうというループを繰り返す。

ルール適用後の状態を、マッチするルール数で評価した理由は、各ルールが部分レイアウトの悪状態を示しているためである。つまりマッチするルールが多いほど悪レイアウト箇所が多く存在することになる。ルール適用後にマッチするルールが増加する場合は、そのルールを適用した結果、周囲に対して悪影響を与え、ルール適用前より悪レイアウト箇所が増加していると解釈できる。上の推論機構によりそのような場合はそのルールは採用されず、他の適用候補ルールについて同様な評価を行ない、マッチするルールが減少するようなルールのみが採用される。

４．評価結果

大型計算機上のCommon Lispを用いてプロトタイプシステムを作成し、評価実験を行なった。ベンチマークとして、１６ビットＡ／Ｄコンバータの部分回路（７１素子、５１ネット）を用いた。改善結果を図４．１に示す。（ａ）が初期レイアウト、（ｂ）が改善結果である。初期レイアウトは自動レイアウトプログラムで生成したもので、制約は全て満足している。

改善処理によりレイアウト制約に違反することなく、初期レイアウトの面積を７％縮小できた。

用いた改善ルールを以下に示す。

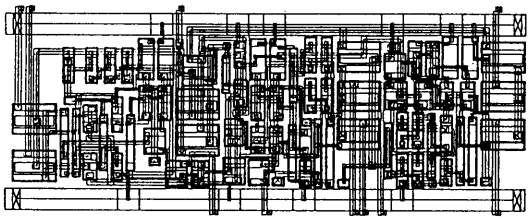
＜面積縮小＞

①素子の配置交換による配線の短縮。
②配線の径路変更による空き領域確保。
③空き領域への素子の移動による詰め合わせ。
④素子の移動に伴う配線の整形。
⑤配線のＡ１２層からＡ１１層への変換。
⑥スルーホールの除去と生成。

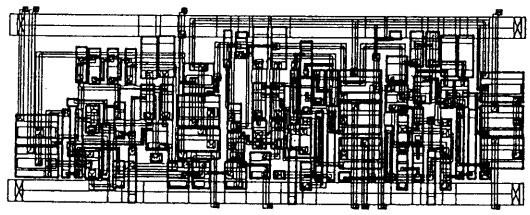
＜制約＞

①素子の配置交換によるペア素子隣接配置制約。
②配線の径路変更による交差、並走禁止制約。

現状では、まだルールの改善処理機能が十分でないため、制約遵守ルールを制約違反修正に用いていない。しかし、予測推論機構によって、制約違反を生じさせるルールが選択されるのを妨げる働きをしている。今後は、制約遵守ルールの改善処理機能を充実させて、制約違反修正に用いられるようにする。



（ａ）初期レイアウト



（ｂ）改善後レイアウト

図 5.1　改善結果

５．おわりに

以上、知識処理を応用したアナログＬＳＩのためのレイアウト手法について、その基本概念、処理方式、評価結果について報告した。

本稿で提案した手法は、レイアウト改善に知識処理を応用し、レイアウト制約違反を修正するルールと面積を縮小するルールをルールベースに用意しておき、レイアウトとルールの照合、実行を繰り返すことにより改善を行なうものである。そのための知識処理方式として、動的集合と予測推論方式について述べた。

プロトタイプシステムにより評価を行なった結果、レイアウト制約違反なく初期レイアウトの面積を７％縮小できた。

参考文献

[１] Kozawa,T.,et al.“Automatic placement algorithms for high packing density VLSI,” Proc. 20th DAC.,pp175-181,1983

[２] Winner.,et al.,“Analog Macrocell Assembler” VLSI System Design May 4.,pp68-71,1987

[３] Kayal,M.,et al.,“SLAIM:A Layout Generation Tool for Analog ICs,” Proc.IEEE Custom Integlated Circuit Conf.,1988

[４] Rijmenants,J.et al.,“ILAC:An Automated Layout Tool for Analog CMOS Circuits,” Proc. IEEE Custom Integlated Circuit Conf.,1988

[５] Garrod,David J.,et al.,“Automatic Layout of Custom Analog Cells in ANAGRAM,” Proc. 22nd DAC.,pp266-272,1985