設計自動化 62－41
(1992. 5.29)

既存のレイアウトからのレイアウト記述の自動生成

重弘 裕二　白川 功　長尾 明†　神戸 尚志†

大阪大学 工学部　†シャープ株式会社

あらまし　機能セルをレイアウト設計する場合，設計済みのレイアウトパターンをいかにして再利用するかという問題が実用上極めて重要である．しかしながら，既存のデータは座標値などが具体的な数値で表現されているため，目的に合わせてデータを更新することは極めて煩わしい作業となる．そこで，これに対処するため，本文では既存のデータをレイアウト記述に自動変換する手法について考察する．生成されたレイアウト記述は，基本的なレイアウト情報をパラメータ化させているため，設計変更条件を満たす数値を与えるだけで，必要な機能セルを自動生成できる．

A Recycling Scheme for Layout Patterns Once Generated for Fabrication Technology

Yuji SHIGEHIRO　Isao SHIRAKAWA　Akira NAGAO†　Takashi KAMBE†

Faculty of Engineering, Osaka University
Suita-shi, 565 Japan

†SHARP Corporation
Tenri-shi, 632 Japan

Abstract　In layout design for functional cells, of practical importance is the problem of how to make the best use of those layout patterns which have been already used in an old fabrication technology, especially in terms of recycling design resources. However, each layout element in such layout data is generally expressed solely with the use of coordinate values, and hence it is extremely tedious to update them so as to be used in accordance with a new fabrication technology. To cope with this problem, the present paper describes an automatic recycling scheme for layout patterns once generated for a fabrication technology. This scheme is implemented mainly by means of reproducing layout data in parametric form with the use of a layout description language. Since updated layout data are parametrized, layout patterns of each functional cell can be obtained simply by tuning up parameters in accordance with a new set of design rules. A part of implementation results are also shown.

1. まえがき

集積回路技術の進歩に伴い，VLSI はますます大規模化し，もはや人手だけに頼る設計は不可能となっている．これに対処する計算機援用設計算法に関する研究も進み，各設計工程における自動化が急速に進行している．レイアウト設計は，VLSI 設計の全工程の中で依然としてかなりの部分を占めているが，その工程は基本的には機能セルの設計とそれらの配置配線設計からなる．セル間の配置配線に関しては，これまで多くの研究が進められ，かなり実用化が達成されている [1]．最近では機能セルの設計自動化の研究開発に多くの関心がよせられている．これは，主として以下の事項に起因するものと思われる．

(i) VLSI 製造技術の急速な進歩によりプロセス技術に適合したセルの再設計が頻繁になったため，多大な手間を要するようになったこと．

(ii) サブミクロンプロセスでは，配線遅延が回路動作に大きな影響を及ぼすため，機能セルの出力信号駆動能力の最適化が新たな問題となっていること．

(iii) 大規模 VLSI を短期間に開発するためには，CPU，PLA，ROM，RAM などの中規模以上の機能セルライブラリの拡充が重要となっていること．

機能セルを人手設計する場合，それまでに設計済みのものをできるだけ再利用する方法が一般的である．なぜなら，回路構造に基本的な変化は少なく，製造技術の変化による設計ルール変更への対応，利用目的に合わせた特性への対応が主たる設計変更の対象となるからである．ところが，人手設計においてよく用いられているグラフィック端末を利用した対話型レイアウトエディタでは，座標値などのデータが具体的な数値で表現されているため，設計規則や素子サイズの変更に対応してデータを更新することは極めて煩わしい作業となっている．このような設計変更を自動的に実行するために，まず，機能セルのレイアウトパターンをレイアウト記述に自動変換し，レイアウト記述中の基本的なレイアウト情報(長さ，幅，相対位置等)をパラメータ化し，次いで，パラメータ化されたレイアウト記述に設計変更条件を満たす数値を与えることにより，要求された機能セルを自動生成する手法が考えられる．

本文では，機能セルのレイアウトパターンをレイアウト記述に自動変換する手法について考察する．本手法では，まず，機能セルのレイアウトパターンからレイアウト情報を抽出し，レイアウト記述言語を用いてパラメータ化する．このように既設計のレイアウトパターンをレイアウト記述に変換しておけば，設計変更条件を満たす数値をパラメータとして与えるだけで，要求された機能セルを自動生成できる．また，新しいレイアウト記述が必要な場合にも，グラフィックエディタを利用してレイアウトパターンを作成し，それを本手法によりレイアウト記述に自動変換することにより，これまでと同等の手間でレイアウト記述を得ることができる．

従来，アナログ回路を対象としたものではあるが，同様の試みが報告されている [9]．この手法では，素子の記述とネット情報に関する記述のみが自動生成される．記述からレイアウトパターンを再生成する際には，まず素子の位置が記述にしたがって決められ，その後，自動配線手法により素子間配線が行なわれる．このため，素子と配線が複雑に入り組んでいるディジタル回路に対していかにして適用するかという問題が残る．本手法では配線の記述も自動生成するので，このような問題は生じない．

以下，2. では，処理対象となるマスクレイアウトとその記述について述べ，3. では，処理手法について考察する．さらに，4. では，セルの再生成例により本手法の有効性を示し，5. では，結論および今後の課題を述べる．

2. マスクレイアウトとその記述

本文では，既存のレイアウトデータをレイアウト記述に変換する手法について述べるが，その最終目的は，設計規則の変更によって素子サイズの修正が必要となった場合に，そのレイアウト記述に基づいて必要なレイアウトデータを再生成することにある (図 1)．以下，レイアウト記述に変換される既存のレイアウトを**入力レイアウト**，レイアウト記述から再生成されるレイアウトを**出力レイアウト**と呼ぶ．本章では，入力レイアウトおよびレイアウト記述について述べ，出力レイアウトについて考察する．

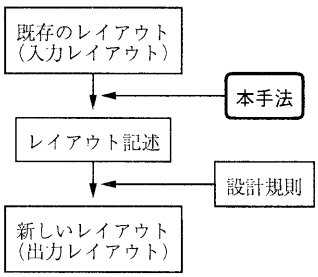


図 1: レイアウト処理系

2.1 入力レイアウト

レイアウトデータとは，IC 内部のシリコンウェハ上に形成すべき酸化膜層やポリシリコン層などのレイアウト要素の形状を表すデータの集合体である．各層のレイアウト要素は矩形や円などの単純な図形を組み合わせた形をしているため，レイアウトデータを，層情報を持つ単純な図形(以下，**プリミティブ** (primitive) と呼ぶ) の集合体とみなすこともできる．通常，レイアウトを設計する際には，まず，トランジスタなどのような，電気的なある一つの回路機能を持つように配置されたいくつかのプリミティブの集合体(以下，**コンポーネント** (component) と呼ぶ) をレイアウト要素として定義する．次に，回路機能を持つように配置されたいくつかのコンポーネントの集合体(以下，**セル**(cell) と呼ぶ) を，さらにレイアウト要素として定義する．大規模なレイアウトを設計する際には，セルやコンポーネントの集合体を，さらにセルと見なすこともある．最後に，IC の全回路を実現するようにセルやコンポーネントを配置することにより，IC 全体のレイアウトデータを完成する．

本手法を用いたレイアウト再生成システムでは，出力レイアウトが入力レイアウトと大きく異なることはない．したがって，出力レイアウトの品質は入力レイアウトのそれとは本質的に同等である．そのため，本手法では，人手で入力され実際の利用に耐え得る品質を持ったレイアウトを処理の対象とする．すなわち，入力レイアウトが以下の条件を満たすことを想定している．

(1) 設計規則に違反していないこと．

(2) 必要に応じて配線に折れ曲がり (jog) を挿入するなど，セル面積を最小化するための種々のレイアウト処理が施されていること．

さらに，問題を単純化するために入力レイアウトを以下のように限定する．

(3) プリミティブの形状は，各辺がチップの底辺に対して水平もしくは垂直な矩形のみとする．

(4) コンポーネントは，トランジスタ (transistor)，コンタクト (contact)，配線 (wire)，およびウェル (well) の 4 種類のみとする．

2.2 レイアウト記述

レイアウト記述とは，言語により表現したレイアウトを意味する．言語を用いることにより，変数や制御文を用いた汎用性の高い表現が可能となり，設計規則や素子サイズの変更を容易化できる．

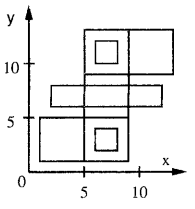
レイアウトの記述に用いる言語を**レイアウト記述用言語**と呼ぶ．レイアウト記述用言語には，レイアウト要素の相対的な位置の関係を指定する形式のもの [2, 4, 6] と，レイアウト要素の座標を具体的に指定する形式のもの [3, 5, 7, 8] がある．前者は，レイアウト要素の座標値の決定を処理系が行うため設計者の負担は軽減されるが，座標を細かく指定することが困難となる．

本処理系ではレイアウト記述を自動生成することを前提としているため，設計者の負担について考慮する必要がない．したがって，本文では座標を細かく指定できる後者のレイアウト記述用言語を対象とする．言語処理系の実現の容易さを検討した結果，**4.** では C 言語とレイアウト記述用の関数を用いたレイアウト記述手法 [7] を用いて実験を行なった．以下，本文でも手法 [7] を用いてレイアウト記述を例示する．

手法 [7] では，C 言語とレイアウト記述用の関数群によりレイアウトの記述が行えるので，C 言語の制御文や演算子をそのまま利用できる．本手法で自動生成されるレイアウト記述では，レイアウトはプリミティブの集合体として表現されるため，プリミティブ記述用の関数群を用意した．例えば，メタル層のプリミティブを

m_rectangle(*x1*, *y1*, *x2*, *y2*);

と記述することにより，引数によるプリミティブの頂点の X 座標，Y 座標の指定を行なうことができる．必要となるすべての層に対し同様の関数を用意した．これらの関数群により，例えば図 2 (a) のようなレイアウトは図 2 (b) のように記述される．



(a) レイアウト例

```
gp_rectangle(2,6,12,8);
d_rectangle(5,1,9,13);
m_rectangle(1,1,9,5);
m_rectangle(5,9,13,13);
c_nc_rectangle(6,2,8,4);
c_nc_rectangle(6,10,
                8,12);
```

(b) 記述例

図 2: レイアウト記述例

2.3 出力レイアウト

本手法の最終目的はレイアウトデータを再生成することにある．出力レイアウトは，設計規則に違反していないのは当然として，面積が小さく，電気的特性が良いものでなければならない．本手法は以下のような特徴を持つため，入力レイアウトと同程度の品質を持つ出力レイアウトを再生成することができる．

(1) 入力レイアウトにおいて大きさが異なるレイアウト要素は，出力レイアウトにおいても異なる大きさに指定できる．したがって，速度や集積度を向上させるために，異なる太さの配線やゲート幅の異なるトランジスタを使い分けている入力レイアウトから，同等の出力レイアウトを得ることができる．

(2) 出力レイアウトにおいて配線長が不必要に長くならない．したがって，配線長の増加に基づく電気抵抗の増加や動作速度の低下を避けることができる．

(3) 入力レイアウトにおいて存在しないプリミティブは，出力レイアウトに追加されない．電気的な接続関係を保つために，必要があればプリミティブが伸長する．

例えば人手設計されたレイアウトでは，隣接によりレイアウト要素を接続することがある．図 3 (a) のレイアウトに対しプリミティブ間の分離幅を増加させるような設計規則の変更が生じた場合，図 3 (b) のように電気的な接続関係が保てなくなることがある．図 3 (c) のように配線素を追加して対処すると，その箇所の電気抵抗の増加により動作速度の低下を引き起こす可能性があるので，本手法では図 3 (d) のようにプリミティブを伸長させ，なるべく動作速度に影響を与えないようにする．

(4) 入力レイアウトにおいて共有部分を持つ同層のプリミティブは，出力レイアウトにおいても共有部分を持つ．

例えば図 4 (a) におけるプリミティブ A と B のように，近接して配置されているプリミティブ間には低抵抗な電気的接続が要求されている可能性があるため，本手法では，図 4 (b) のように出力レイアウトでも離れないようにする．

3. 処理手法

IC のマスクレイアウト面積はできる限り小さな方が望ましいが，レイアウト要素の大きさの下限やレイアウト要素間の距離の下限を定めた設計規則を犯してはならない．したがって，レイアウト要素の位置は，設計規則と周辺のレイアウト要素の位置によって表される制約条件を満たすように決められている．あるレイアウト要素の座標を a，その周辺にあるレイアウト要素の座標を b とし，その座標間の距離の下限が r という設計規則で与えられるとすると，$b < a$ のとき制約条件は $b + r \leq a$ という式で表される．レイアウト記述において，設計規則に関わるレイアウトパラメータ，素子サイズ，およびレイアウト要素の座標を変数に割り当て，それらの変数を含む式で前述の制約条件を表現し，その式を用いてレイアウト要素の位置を記述することができれば，そのレイアウト記述は，設計規則や素子サイズに依存しないものとなる．例えば，図 5 (b) は図 5 (a) のレイアウトの記述例であるが，設計規則 (sep) や素子サイズ (wid, len) が変数で記述されているため，それらの具体的な数値には依存しない．

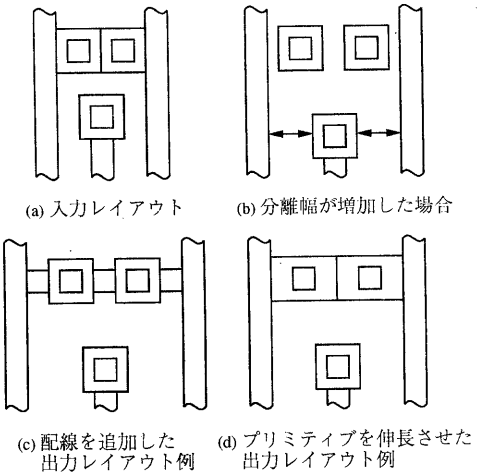


図 3: 隣接による接続

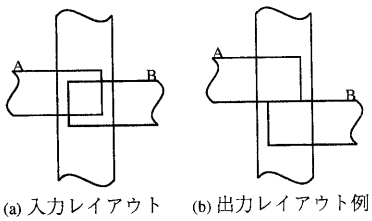


図 4: 重なったプリミティブ

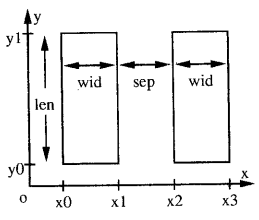
レイアウト要素の座標を節点に，座標間の制約条件を有向枝に対応させたグラフを制約グラフと呼ぶが [10]，本手法ではこの制約グラフを利用し処理を行なう．通常のコンパクション処理では，制約条件が座標値の差の下限として具体的な数値で与えられるため，その値を重みとして枝に付加する．しかし本手法では，レイアウト記述を生成する時点では設計規則や素子サイズが確定できないため，制約条件を具体的な数値で表すことができない．そこで，制約条件自体を属性として枝に付加することとする．処理の概要を以下に示す．

1° 入力レイアウトであるセルのデータの階層構造を展開する．(以後，データはプリミティブ，すなわち矩形の集合体として扱うが，必要に応じて，コンポーネントやセルのデータをも参照する．)

2° **3.1** に示す手順により制約グラフを構築する．

3° そのグラフから冗長な節点や枝を除去する．

4° **3.2** に示す手順によりレイアウト記述を生成する．



(a) レイアウト例

```
x1=x0+wid;
x2=x1+sep;
x3=x2+wid;
y1=y0+len;
m_rectangle(x0,y0,
            x1,y1);
m_rectangle(x2,y0,
            x3,y1);
```

(b) 記述例

図 5: 設計規則に依存しないレイアウト記述

3.1 制約グラフの構築

プリミティブの頂点の座標を節点に，その座標間の制約条件を有向枝に対応させた制約グラフを，X 方向と Y 方向に対してそれぞれ構築する．抽出する制約条件の種類を以下に示す．

(1) プリミティブ間の最小分離幅を維持する (図 6 A)．

(2) プリミティブの最小幅を維持する (図 6 B)．

(3) プリミティブ間のオーバーラップ幅を維持する (図 6 C)．

さらに，**2.3** で述べた特徴を実現するために，以下の条件も抽出する．

(4) 重なっているプリミティブの辺の順序を入れ換えない (図 6 D)．

(5) 配線を構成するプリミティブの長さを可能な限り短くする (図 6 E)．

抽出した条件の種類や，適用される設計規則の種類は，**3.2** に示す手順において必要となるので，有向枝に属性として付加しておく．

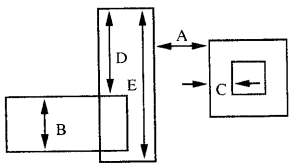
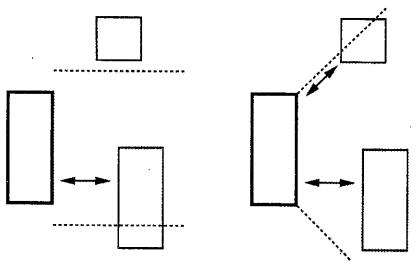


図 6: 制約条件

(2), (5) は，プリミティブ内の向かいあった辺の座標の間に存在する条件であり，個々のプリミティブを調べれば容易に抽出できる．また，(3), (4) は，重なり合うプリミティブの辺の座標の間に存在する条件であり，重なり合うプリミティブを調べれば容易に抽出できる．(1) は，レイアウトのコンパクション処理においてよく利用されるため，抽出法についても多くの研究がなされているが [11]，本手法では，これまでの抽出手法をそのままでは利用できない．これは，通常のコンパクション処理では必要とならない制約も必要となることがあるためである．これを解決するため shadow-propagation 手法 [11] を修正した手法により制約グラフを構築する．

制約グラフを用いるコンパクション処理では，まず X 座標に関する制約グラフを作り，これを用いて X 座標を更新した後で，Y 方向に対する処理を行なうために Y 座標に関する制約グラフを作る．したがって，例えば X 座標に対する処理を行なう際には，図 7 (a) に示す範囲のレイアウト要素に対して制約条件を考慮すれば十分である [11]．しかし，本手法では X 座標を更新せずに Y 座標に関する処理も行なわなければならない．なぜならば，設計規則の値や素子サイズが与えられるのはレイアウト記述が生成された後であり，記述生成処理の途中では座標値を更新できないためである．そのため，例えば X 座標に対する処理を行な

う際には，図 7 (b) に示す範囲のレイアウト要素に対して制約条件を考慮する．Y 座標に対しても同様の処理を行なえば，周囲のプリミティブ全てに対して制約条件が考慮されるため，出力レイアウトが必ず設計規則に従うことが保証される．



(a) コンパクション手法　　(b) 本手法

図 7: 制約条件を考慮する範囲

制約条件 (1) を効率良く抽出するために，全てのプリミティブに対して次の処理を行なう．ただし，以下の処理でプリミティブを探索する領域を影と呼ぶ．

1° プリミティブの辺から，45 度の角度で影を広げる (図 8 領域 S)．

2° 影が，他のプリミティブの辺 (1°の辺と平行なもの) に届いたら，その辺の対向辺から，さらに影を広げる (図 8 領域 T)．以下，これを影の影と呼ぶ．

3° 影が届いたプリミティブの辺に，もし影の影が届いていなければ (図 8 矩形 B)，制約グラフに枝を加える．

影の影が届いたプリミティブ (図 8 矩形 C) に対しては影の影を発生したプリミティブ (図 8 矩形 B) との間の制約条件が考慮されるため，影を発生したプリミティブ (図 8 矩形 A) との間の制約条件は冗長となる．

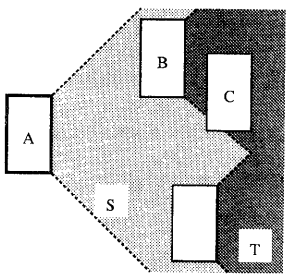


図 8: 影の伝搬

3.2 レイアウト記述の生成

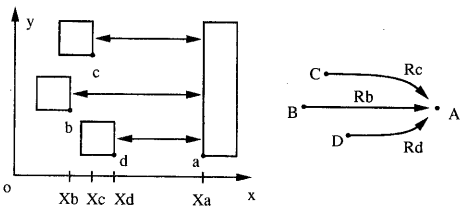
レイアウト記述は，図 5 のように，主に座標値を求めるための記述と矩形の記述からなる．後者は自明であるので以下では前者について考察する．

制約グラフに抽出された制約条件は，プリミティブの頂点が互いにどれだけ離れなければならないかを示している．頂点 a，b，c，d の座標 Xa，Xb，Xc，Xd が制約グラフ上の節点 A，B，C，D にそれぞれ対応し，B，C，D から A に向かって有向枝が出ているとき，各枝の持つ制約条件がそれぞれ Rb，Rc，Rd という変数で表されるならば (図 9)，Xa の値は Xb+Rb，Xc+Rc，Xd+Rd の最大値になるべきである．そこで，Xa に対し

```
Xa = Xb + Rb;
Xa = max(Xa, Xc + Rc);
Xa = max(Xa, Xd + Rd);
```

(max() は，大きな方の引数を返す関数)

のような記述を出力する．左から順にすべての座標について同様に記述を出力すれば，すべての座標を求めるためのレイアウト記述が得られる．ただし，このような記述から得られる出力レイアウトはプリミティブが左に詰められたものとなり，配線が不必要に長くなってしまうことがある．



(a) レイアウト例　　(b) 制約グラフ

図 9: 複数の制約条件

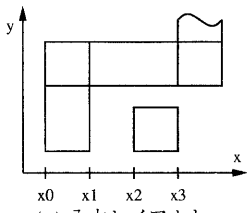
通常のコンパクション処理において，制約グラフに基づいてレイアウト要素の座標を決定するには，まず制約グラフのクリティカルパスを求め，その経路上にある節点に対応する座標の値を決定し，その後で，配線長を考慮の上，残りの座標の値を決定する [11]．しかし，この手順は本手法には適用できない．なぜなら，レイアウト記述を生成する時点では制約グラフの枝の重みの値を確定できず，クリティカルパスを特定できないからである．

本手法では，すべてのプリミティブの頂点の座標値を計算するための記述を出力した後で，出力レイアウトにおいて配線長を必要最小限にするために，座標値を再計算するための記述を繰り返し出力する．例えば図 10 (a) に示す入力レイアウトに本手法を適用すると，まず，図 10 (b) に示すように (記述とその記述から再生成される出力レイアウトを同時に示す) 左から順にすべての座標について記述を出力し，次に，図 10 (c) に示すように右から順に一部の座標についての記述を再度出力する．さらに，配線長を必要最小限にするために必要に応じて同様な記述の出力を繰り返し，最終的なレイアウト記述が得られる．記述を繰り返し出力する座標は，条件の種類や周囲のプリミティブの座標の出力状況をもとに選択する．

4. 実験結果

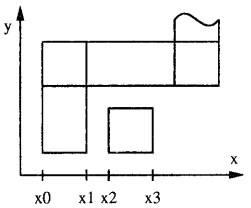
本手法，ならびにレイアウト記述用言語処理系を UNIX ワークステーション上で C 言語を用いて実現し，実際のレイアウトに対して実験を行なった．

まず，本手法を用いて図 11 (a) のレイアウトからレイアウト記述を自動生成した．その際，処理対象としてスタンダードセルを想定し，上半分と下半分を別々に処理した．レイアウトに含まれる矩形の数は 153 であり，生成された記述に含まれる座標値を計算するための式の数は 2839 であ



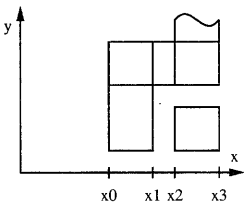
(a) 入力レイアウト

```
x0 = ···;
x1 = x0 + wid;
x2 = x1 + sep;
x3 = x2 + wid;
···
```



(b) レイアウト記述と出力レイアウト (1)

```
x0 = ···;
x1 = x0 + wid;
x2 = x1 + sep;
x3 = x2 + wid;
···
x3 = ···;
x2 = x3 - wid;
x1 = x2 - sep;
x0 = x1 - wid;
···
```



(c) レイアウト記述と出力レイアウト (2)

図 10: レイアウト記述の生成

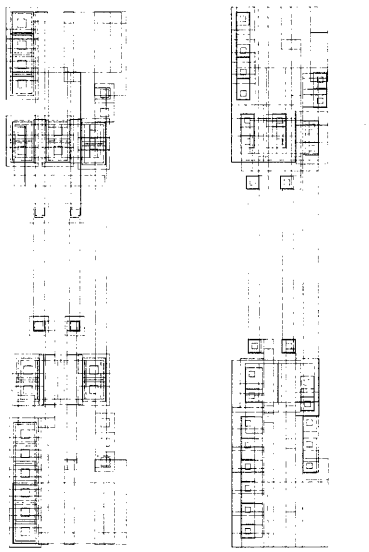
る．記述の生成処理には 12 秒を要した (SPARC station 2 を使用)．

さらに言語処理系を用いて，その記述から図 11 (a) とは異なる設計規則，素子サイズのレイアウトを生成した．そのレイアウトを図 11 (b) に示す．

5. むすび

本文では，機能セルのレイアウトパターンをレイアウト記述に自動変換する手法について述べた．本手法によって得られたレイアウト記述では設計規則や素子サイズが変数で表現されているため，それらの値をパラメータとして与えるだけで利用目的にあったレイアウトを生成できる．従来，このようなレイアウト記述が必要な場合にはテキストエディタによる人手設計に頼らざるをえなかったが，本手法を活用すれば，グラフィックエディタを利用した設計や既存のレイアウトの再利用などが可能となる．

問題点としては，斜め向きの矩形が扱えないこと，レイアウトの階層構造や繰り返し構造が記述に反映されないため人手による設計に比べレイアウト記述の量が多いこと，などがあげられる．これらを解決するためには，レイアウトから情報を抽出する能力の強化と，抽出した情報を効率良く表現する記述手法の開発が必要であり，今後の研究課題である．



(a) 入力レイアウト　(b) 出力レイアウト

図 11: 実験結果

参考文献

[1] A. Sangiovanni-Vincentelli: "Automatic layout of integrated circuits", in Design Systems for VLSI Circuits, ed. G. D. Micheli, pp.113-196, Martinus Nijhoff Publishers (1987).

[2] R. J. Lipton, S. C. North, R. Sedgewick, J. Valdes and G. Vijayan: "ALI: A procedural language to describe VLSI layouts", Proc. 19th DAC., pp.467-474 (1982).

[3] P. A. D. Powell and M. I. Elmasry: "The icewater language and interpreter", Proc. 21st DAC., pp.98-102 (1984).

[4] J. M. Mata: "ALLENDE: A procedural language for the hierarchical specification of VLSI layouts", Proc. 22nd DAC., pp.183-189 (1985).

[5] W. E. Cory: "Layla: A VLSI layout language", Proc. 22nd DAC., pp.245-251 (1985).

[6] J. A. Solworth: "GENERIC: A silicon compiler support language", Proc. 23rd DAC., pp.524-530 (1986).

[7] 原嶋勝美, 若林謙二, 神戸尚志, 重弘裕二, 白川功: "C言語によるレイアウト記述の一手法", 信学技報, **CAS87**-14 (1987-05).

[8] 神原弘之, 小野寺秀俊, 田丸啓吉: "アナログモジュール自動生成のためのレイアウト記述の一手法", 1989 信学春季全大, SA-7-1 (1989).

[9] 佐藤寿倫, 小野寺秀俊, 田丸啓吉: "モジュール・ジェネレータ開発容易化の一手法", 信学技報, **CAS90**-100 (1990-11)

[10] Y. Z. Liao and C. K. Wong: "An algorithm to compact a VLSI symbolic layout with mixed constraints", IEEE Trans. Computer-Aided Design, **CAD-2**, 2, pp. 62-69 (1983).

[11] D. G. Boyer: "Symbolic layout compaction review", Proc. 25th DAC., pp.383-389 (1988)

盛光印刷所