設 計 自 動 化　43－4
（1988. 7. 12）

# 任意角度辺を含むLSIマスクパターンの拡大･縮小手法

長尾 明、小嶋 格、神戸 尚志

シャープ株式会社　コンピュータシステム研究所

リサイジングは図形演算のひとつであり、マスクパターンの拡大や縮小を行う操作のことである。マスクパターンを構成する図形のほとんどは、垂直･水平線分からなっているが、高集積化の実現や、回路特性の向上のために斜め辺も用いられている。

このようなマスクパターンをリサイジングできるよう、任意角度辺からなる図形を対象とし、かつ高速に処理できる手法を開発したので報告する。本手法は、リサイジング図形を得る前にいったん「仮図形」と呼ぶ中間的な頂点列を生成することにより、アルゴリズムの簡単化と高速化を図っている。

本稿ではまず、入力図形とリサイジング図形を比較し、その変化分の図形について考察するとともに、仮図形とはどのようなものであるかを示し、次いで仮図形の生成法を含めたリサイジングのアルゴリズムについて述べている。最後に、本手法が頂点数nのマスクパターンを、O(n log n)の計算複雑度で高速に処理することを示す。

# A RESIZING ALGORITHM FOR THE LSI LAYOUT WITH DIAGONAL EDGES

Akira NAGAO,　Itaru KOJIMA,　Takashi KAMBE

Computer Systems Laboratories,　SHARP Corporation

2613-1　Ichinomoto-cho,　Tenri-shi　Nara　632　Japan

The resizing operation which is one of the pattern operation for the LSI layout, expands or shrinks it. The LSI layout pattern includes a lot of othogonal figures, but diagonal figures are often used in order to accomplish a high density layout and to improve a characteristic of a circuits.

We have developed a high-speed resizing algorithm for the LSI layout with diagonal figures using the intermediate figure called "imaginary figure". In this paper, we discuss the difference of an objective figure and the figure given by the resizing, and describe "what is the imaginary figure ?" Next, we show the making of the imaginary figure, and the flow of the resizing algorithm. At last, it is shown that the resizing algorithm manipulates the LSI layout pattern with n-vertices in O( n log n) time complexity.

## 1. はじめに

LSIの大規模高集積化に伴い、今やCADシステムの支援なくしてLSIを設計することは、不可能となっている。LSIの設計では、各設計段階に応じたCADシステムが利用されているが、なかでもマスクパターンを対象としたアートワークシステムにおいて、図形演算機能はなくてはならないものである。リサイジング(resizing)は図形演算のひとつで、マスクパターンの拡大(expanding)あるいは縮小(shrinking)を行うものであり、マスクパターンの検証やPG,EBデータの作成に利用されている。ここで云う拡大･縮小とは一般に、マスクパターンを入力図形とし、その各辺を図形の外部あるいは内部へ指定された変更幅だけ平行移動して、拡大図形･縮小図形を得る操作のことである。

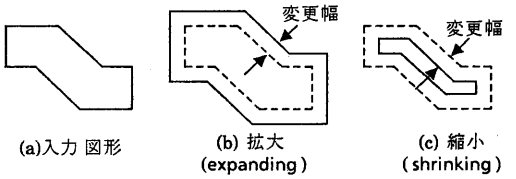



図1　リサイジング機能

リサイジング手法としては、水平･垂直辺からなる図形を対象とし、平面掃引を用いて高速に処理するXY法が知られており、45°･135°の斜め辺を含む図形も扱えるような拡張も報告されている。[1]

しかし近年、マスクパターンに60°･120°等の斜め配線も使われるようになり、また円弧を多角形に近似する場合、多角形の頂点数によりその辺の角度は様々に変化する。そこで、任意角度辺を含む図形を高速に処理できるリサイジング手法が必要となっている。

本稿で提案する手法は、まず入力図形であるマスクパターンの辺を変更幅だけ平行移動し、この辺にもとづいて中間的な図形(仮図形)を得、次にこの図形をもとにリサイジング図形を得るというものであり、任意角度辺を含む図形を高速に処理することができる。

本稿では最初に、リサイジングの前後における変化分の図形について述べ、ついで仮図形を導入すればリサイジング図形が正しく得られることを示した上で、仮図形を用いたリサイジング手法を提案する。最後に、この手法を用いると、入力図形の頂点数nに対しO(n log n)の計算複雑度で処理ができることを示す。

## 2. リサイジングとリサイジング図形

リサイジングのアルゴリズムについて述べる前に、リサイジングとはどのようなものであるか考察する。

### 2.1 入力図形

まず、リサイジングの対象となる入力図形の条件を以下に示す。

(1) 任意角度辺からなる多角形である。
(2) 図形はあらかじめ輪郭取りされており、重なり部分はない。
(3) 入力図形に限らず、図形の辺はすべてベクタとみなし、その向きはベクタの進行方向に対し図形の内部を右にみる方向にとってある。即ち、頂点は時計回りに順序づけられる。
(4) ドーナツ図形とも呼ばれる窓あき図形も対象とする。尚、窓部分の頂点は負図形であると考えられ、反時計回りに順序づけられる。

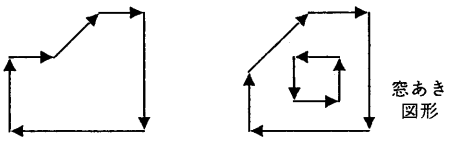



図2　入力図形とベクタ

### 2.2 リサイジング

入力図形と、リサイジングの変更幅dが与えられた。ここで図形の辺は連続する点からなっていると考え、まず辺上にあるすべての点について、これを中心とする半径dの円を考える。円の中心は辺上に連続して無数に存在するから、中心が辺に沿って図形を一周するように移動すれば円もまた連続した軌跡を描く。

拡大とは、入力図形にこのような無数の円を加える操作であると定義する。また縮小とは、このような無数の円を入力図形から削除する操作であると定義する。拡大で加える円を右回りと考えると、縮小とは、左回りの円(負図形の円)を加える操作とみなすこともできる。

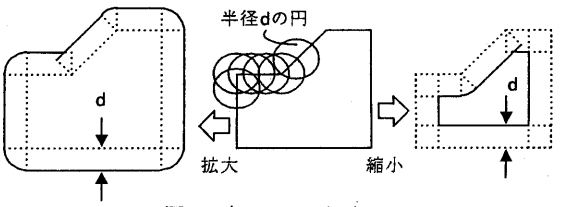



図3　真のリサイジング

### 2.3 リサイジング図形(拡大図形･縮小図形)

こうして得られた図形は円弧を含んでおり、リサイジングにつづいて行われる他のアートワークにとって、むやみにデータ量を増やすばかりでなく、複雑な処理を要求することにもなる。円弧は拡大時には凸角部分に、また縮小時には凹角部分に生じるが、リサイジングを必要とするアートワークにとって、矩形をリサイジングした結果もまた矩形で充分という場合が多く、本稿では図4に示すように、円弧部分を両脇にある辺の延長で近似して、一般的なリサイジングの図形とする。

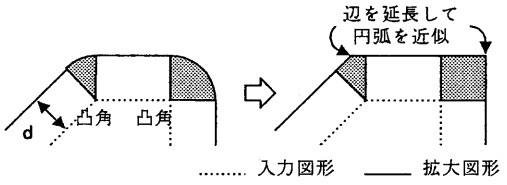



図4　拡大時に生じる円弧とその近似

また円弧を含むリサイジング図形を、一般的なリサイジング図形と区別するために真のリサイジング図形と呼ぶことにする(図5)。本稿では、一般的なリサイジング図形を得る手法について述べ、以後特にことわらない限りリサイジング図形とは、一般的なリサイジング図形を意味するものとする。

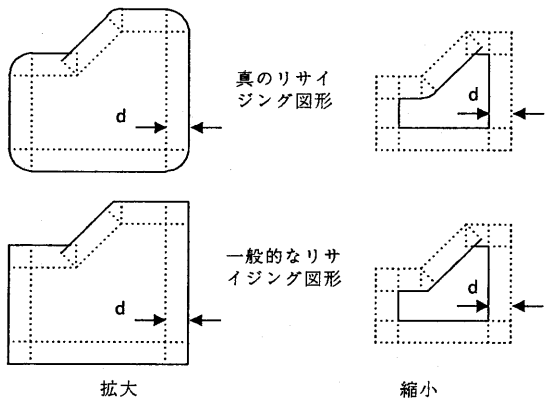



図5　リサイジング図形

一般的なリサイジング図形はおおざっぱな近似であるから、拡大において凸角の角度が小さいほど、また縮小において凹角の角度が大きいほど、真のリサイジング図形とのギャップは大きくなる。このギャップを小さくする手法については4.2節で述べる。

## 3. 仮図形

仮図形とは、本稿で提案するリサイジング手法において必要となる中間的な図形のことである。本章ではまず、一般的なリサイジング図形を得る際の、追加･削除図形について考察し、これをもとに仮図形とはどのようなものであるかを示す。

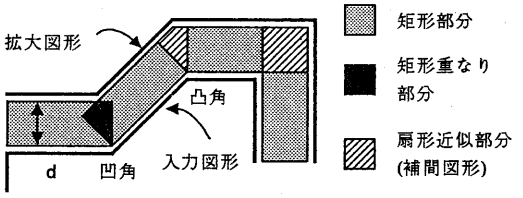



図6　拡大における追加部分

### 3.1 追加･削除図形

**(拡大)**　入力図形と拡大図形を見比べると(図6)、追加部分が以下に示す矩形部分(①)と、扇形を近似した部分(②)とからなっていることがわかる。①矩形部分は一方の辺が入力図形の辺であり、もう一方の辺の長さが変更幅dである。②扇形の近似部分は凸角にだけ生じ、凸角の角度をAとすると、もとの扇形の中心角は(180°－A)、半径はdである。扇形を近似した四角形は、追加する矩形どうしをつなぐ部分であるから、以後これを補間図形と呼ぶことにする。凹角部分に補間図形を生じないのは明らかで、そのかわり矩形の端が互いに重なりあう。

**(縮小)**　入力図形と縮小図形を見比べると、削除部分は拡大と同様に矩形と補間図形とからなっている。ただし補間図形は凹角部分に、矩形の重なりは凸角部分に生じる。

以上のことから、拡大とは矩形と補間図形を入力図形に追加する操作であり、縮小とは矩形と補間図形を入力図形から削除する操作であると考えられる。

このような図形の追加･削除の操作は、図形演算のなかでも一般にパターン論理演算と呼ばれる演算の、OR･SUBにあたる(図7)。このパターン論理演算については、斜め辺を含む図形を高速に処理できるアルゴリズムが知られている[2]。

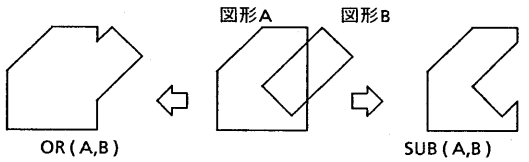



図7　パターン論理演算

追加･削除される図形も、入力図形同様ベクタ表現されるが、縮小における削除図形のベクタ表現を逆向きに(頂点を反時計回りに)とって、負図形としておけば、削除は負図形を追加する(入力図形とこの負図形とでOR演算する)操作に等しい。

ここで云うOR演算とは、ベクタの向きから図形の内外を判断して図形の重なり数[2]を得、重なり数が1以上の部分を取り出すパターン論理演算のことである。通常の図形の内側の重なり数は1、外側は0である。又2つの図形が重なっている部分では2、負図形の重なり数は－1であると考える。(図8)

従ってリサイジングは、入力図形の辺をもとに矩形と補間図形を発生し、入力図形とOR演算を施すことで実現できる。

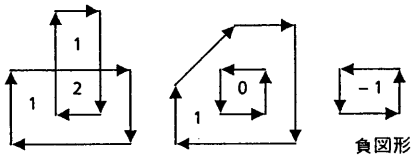



図8　ベクタ表現した図形と重なり数

### 3.2　仮図形

補間図形と矩形、入力図形と矩形は互いに辺を共有しており、ベクタで表現した場合に共有している辺は、互いに逆向きのベクタとなっている。OR演算の内部処理において、このようなベクタは相殺されるので、OR演算の入力として与える図形の中から、共有している辺をあらかじめ省いておけば、OR演算の負荷が軽くなる。相殺される辺は、厳密には次の箇所である。

(1) 補間図形の4辺のうち、扇形の半径にあたる2辺
　これは、となりあう矩形の辺と相殺される。

(2) 入力図形の辺
　これは、この辺をもとに発生した矩形の辺と相殺される。(図9)

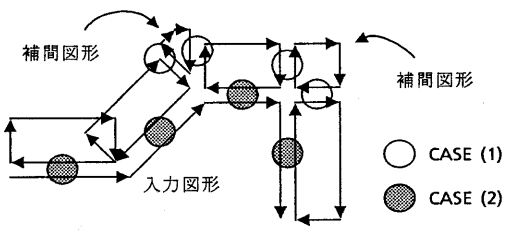



図9　拡大図形とベクタの相殺

このように、共有している辺をベクタと考えて相殺すると、入力図形･矩形･補間図形は、一連のベクタで一筆描きできる。また、

(3) 補間図形の相殺されなかった残りの2辺は、となりあう矩形の辺を延長して得たものであり、ベクタの向きもとなりあう矩形の辺と一致するため一本化できる。(図10)

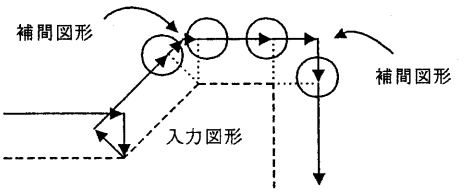



図10　拡大図形とベクタの一本化

こうして得られた一筆描きの図形が仮図形である。このときリサイジングは、入力図形から仮図形を求め、これをOR演算することで実現できる。

仮図形の簡単な例を図11に示す。また付録Aに、拡大･縮小における仮図形の例を、図形の重なり数とともに示す。

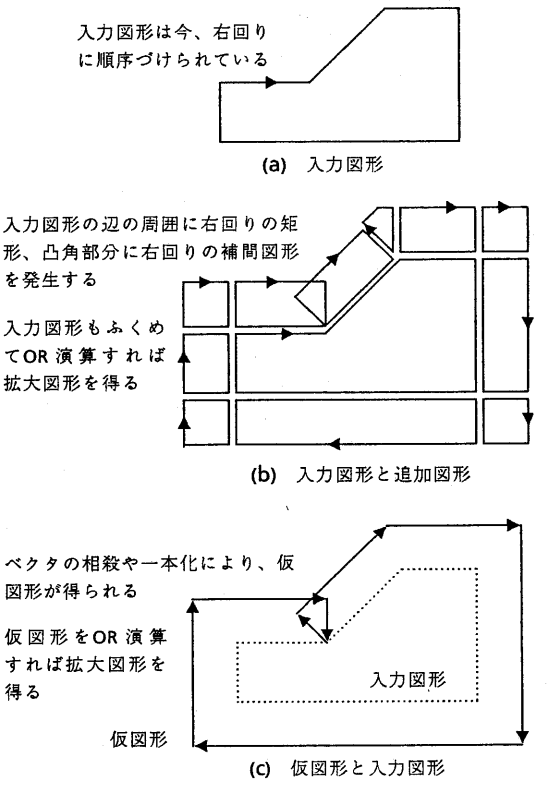



図11　拡大時の仮図形

## 4. 斜め図形のリサイジング手法

3章で示したように、リサイジングは次の2つのステップで実現できる。

STEP 1 仮図形の生成
STEP 2 OR演算

OR演算の手法については高速なアルゴリズムが知られているので、ここでは仮図形の生成手法について述べる。

### 4.1 仮図形の生成法

仮図形はベクタで表現されるが、これを頂点列と考えて、入力図形の頂点列と対応づけることができる。すなわち、補間図形を生じる部分では、入力図形の頂点1つが仮図形の頂点1つに対応し、矩形が重なる部分では、入力図形の頂点1つが仮図形の頂点3つに対応する。

従って仮図形は、入力図形の頂点をベクタの向きに沿って順に注目し、その頂点に対応した仮図形の頂点を求めていくことで得られる。仮図形の個々の頂点座標は、次のように算出される。

拡大(縮小)時に入力図形の頂点Aのなす角度が

CASE 1　凸角(凹角)であれば

補間図形を生じるから、頂点Aを挟む2辺を変更幅d だけ外側へ(内側へ)平行移動し、互いに延長して交点を求める。この交点が仮図形の頂点A'である。(図12)

CASE 2　凹角(凸角)であれば

矩形の重なり部分を生じるから、頂点Aを挟む2辺を変更幅d だけ外側へ(内側へ)平行移動し、頂点Aに対応する側の端点A1',A2'と、もとの頂点Aを仮図形の頂点とする。(図12)

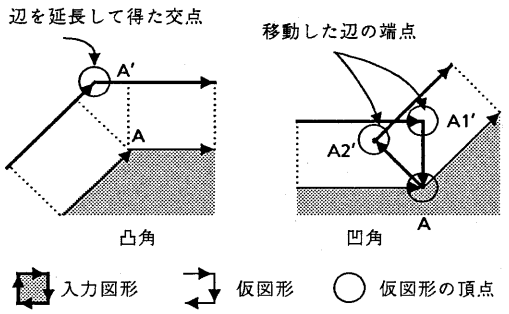



図12　仮図形の頂点生成(拡大)

### 4.2 鋭角の処理

2.3節でもふれたように、拡大で入力図形に角度の小さい凸角があれば、その部分で一般的なリサイジング図形と真のリサイジング図形の間に、大きなギャップを生じる。これは扇形の円弧部分を、2本の辺で近似していることによる。入力図形の頂点の角度をAとして、扇形の中心角は(180° – A)であるから、Aが小さいほど扇形の中心角は大きくなり、円弧部分が広くなって2本の辺で近似するのが難しくなる。縮小で凹角部分に生じる扇形は、中心角が(A – 180°)になることを除き拡大と同様である。

いま1本の辺を追加して、3本の辺で円弧部分を近似することを考える。ここで追加する辺を、円弧の中点を通過する接線上にとれば、真のリサイジング図形とのギャップ面積を小さくできる。このとき生成される仮図形の頂点は、必ず鈍角となるから、追加する辺は1本で十分である。(図13)

このような辺を算出する手法はいくつか考えられるが、ここでは接線上に適当な長さの線分(実際にはベクタ)を発生し、あたかも入力図形の辺を平行移動して得たかのように扱い、その延長線上に交点を求める。

鋭角部分の切り落とし処理を、何度の頂点に適用するかは自由に設定できるが、拡大では90°未満の(縮小では270°を越える)頂点を対象とするのが適当であろう。

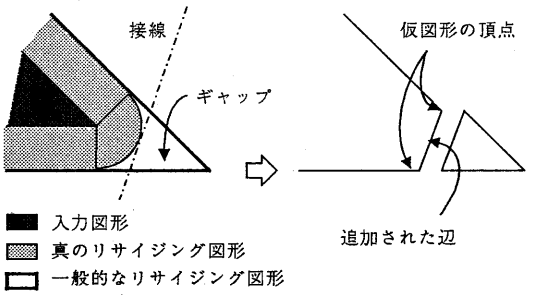



図13　鋭角部分の切り落とし処理

### 4.3 処理手順

仮図形の生成を行うSTEP 1 は、さらに①辺の平行移動と入力図形の頂点の凹凸判定と、②仮図形の頂点生成とにわけて処理できる。後者の基本的な処理は、CASE 1の交点計算と、CASE 1,2で共通な仮図形の頂点座標の登録だけである。

リサイジングの前処理も含めた処理手順を、図14に示す。STEP 1 は①②とも単純な処理で実現でき、STEP 2 は既存の図形演算を利用できるので、全体を通じて単純なアルゴリズムとなっている。付録Bに本手法によるリサイジング過程を示す。

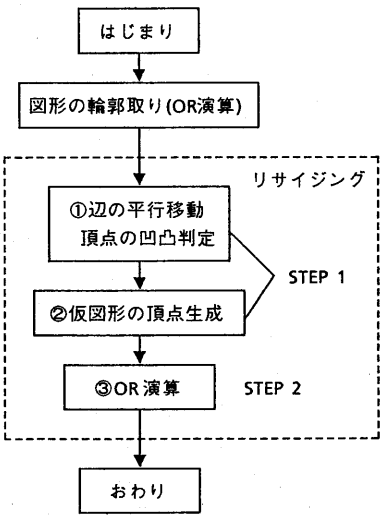



図14　リサイジングの処理手順

### 4.4　本手法の計算複雑度

入力図形の頂点数をnとする。①②の手続きは、入力図形の頂点を辺に沿って注目し、一周すれば終わるからO(n)の手数で処理できる。またこうして得られた頂点数Nの仮図形に対して、③のOR演算はO(N log N) で処理できる[2]。ここで入力図形の頂点1つにつき仮図形の頂点が1~3つ対応するから、Nが3nを越えることはない。従って③の手続きはO(n log n)でおさえられる。以上より、本リサイジング手法の計算複雑度はO(n log n)である。

また一般にリサイジングのアルゴリズムは、変更幅dに対し、拡大時に間隔2d以下の部分で図形が接合してひとつとなり、縮小時に幅2d以下のくびれ部分で図形が分離することを考慮しなくてはならない。入力図形の辺の平行移動とこの接合·分離の判定を、平面掃引で同時に処理するには、スリットの幅を少なくとも2dとる必要があり、dが大きくなるとスリットにかかる辺の数も増加する。

本手法では接合·分離の処理を、STEP 2 の平面掃引を用いたOR演算で処理するが、辺の平行移動は予めSTEP 1 で行う為、スリット幅は変更幅dに関係なく、OR演算が高速に処理できる。

## 5.　単純なリサイジング手法とその問題点

本章では、仮図形の生成方法をさらに簡単化した単純なリサイジング手法について述べる。この手法は、垂直·水平辺だけからなる図形に対して有効であり、斜め辺を含む図形を扱うには、前章で提案した手法が必要である。

斜め図形のリサイジング手法は、入力図形の頂点が凹角か凸角かによって処理を異にしている。この2つのCASEのうちCASE2は、入力図形の1頂点に対応する仮図形の頂点が3つもあるという点において、無駄が多いと考えられる。

そこで凹角凸角に関係なく、すべてCASE 1で処理することを考える。すなわち、平行移動した2辺を延長して交点を求め、これを新たな頂点とすれば、簡単に中間的な図形を得ることができる。こうして得られた図形は、仮図形から図12のCASE 2に示される自己交差部分を省略したものであると考えられ、図12を見る限りでは有効な手段である。

しかし斜め辺を含む図形に対しては、交点を求めて仮図形の3頂点のかわりとする操作が、常に自己交差部分の省略になるとは限らない。付録Bはその一例で、仮図形の自己交差部分と、交点計算による省略部分とが一致しないために、正しいリサイジング図形を得ていない。従って斜め図形を扱うには、CASE 2は最低限必要な処理である。

## 6.　おわりに

本稿では、リサイジング操作における追加·削除図形に注目し、仮図形なる中間的な頂点列を生成して、これをOR演算するリサイジング手法を提案した。この手法は、任意角度辺を含む頂点数nのマスクパターンを、O(n log n)で高速に処理することができる。

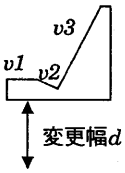
より高速な単純なリサイジング手法は、特殊な例を除いて斜め図形にも有効であるので、今後はこの2つの手法をうまく使いわける工夫が必要である。

## 参考文献

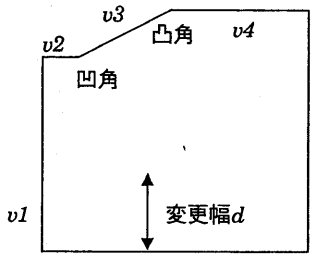
[1] 佐藤 政生、橘 昌良、大附 辰夫、“図形整形アルゴリズムとそのLSIパターン設計への応用”,電子通信学会論文誌(C), '83/12 Vol.J66-C,No.12, pp.1132-1139 (1983)

[2] Tokinori Kozawa、Akira Tsukizoe、Junya Sakemi、Chiei Miura、and Tatsuki Ishii、“A Concurrent Pattern Operation Algorithm for VLSI Mask Data”、IEEE, Proc. of the 18th DAC、pp.563-570 (1981)

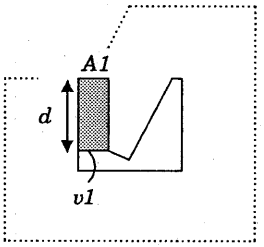
1



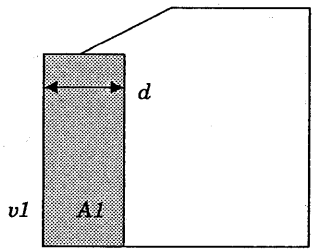

入力図形と変更幅



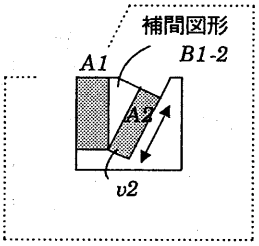

2



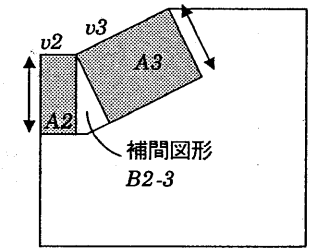

v1をもとに発生した矩形A1
拡大時には右回りの正図形
縮小時には左回りの負図形



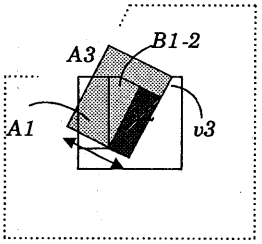

3



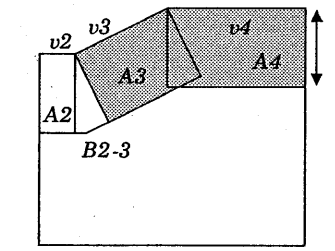

拡大時には凸角、
縮小時には凹角部分に
補間図形を発生する。



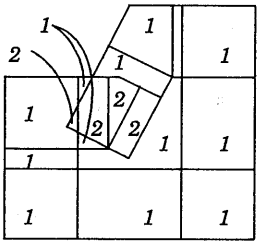

4



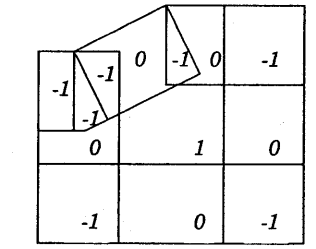

拡大時の凹角、
縮小時の凸角部分には、
補間図形は発生しない。



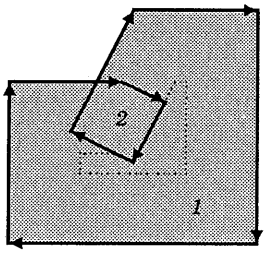

5



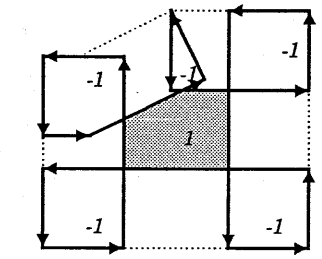
リサイジングの変化分としての矩形及び補間図形、そして入力図形を加えたときの重なり数。このままOR演算してもリサイジング図形は得られる。



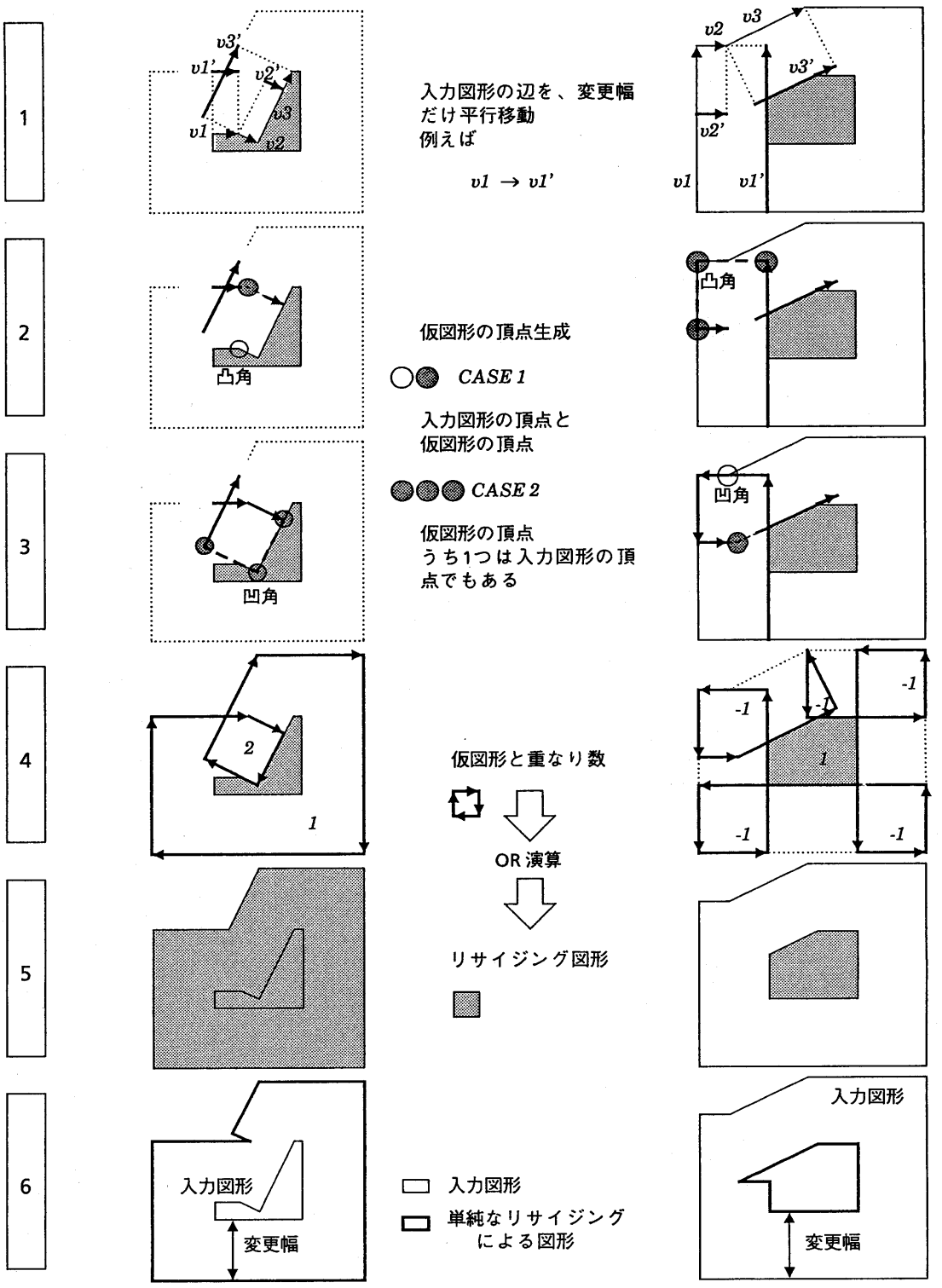
6




仮図形
▨部分は、リサイジング図形。
OR演算を施して得られる。




拡大　　付録A　図形の重なりと仮図形　　縮小




付録B　本リサイジング手法の処理過程