設計自動化30－5
（1986. 1. 21）

ＶＬＳＩブロック寸法最適化配置の階層化

茂垣　真人，　三浦　地平，　寺井　秀一
（日立製作所中央研究所）

1．緒　言

大規模なＶＬＳＩの設計では、論理を幾つかのブロックに別けて設計する階層的設計手法がよくとられる［１］。現在我々は、チップ上のブロック配置の自動化の研究を進めている［２］。

ブロックの形状，大きさはさまざまであり、その配置も不規則である。そのため無効領域ができやすいことが一つの問題である。この問題点を解決する有効な手段として、ブロックの形状を，配置状態に応じて変えるブロック寸法最適化手法を開発した［３］。

しかしこの手法では、内部で線形計画法を使用しているため、メモリ使用量はブロック数の２乗に比例し、処理時間は３乗に比例する。ますます規模の大きくなるＶＬＳＩに対処するためにブロック寸法最適化処理の改良を行い、チップ面積削減効果を保持しつつ、メモリ使用量，処理時間を削減した。

2．従来のブロック寸法最適化処理

2．1　ブロック配置モデル

図１にブロックの配置モデルを示す。矩形のチップ上に互いに重ならないように矩形のブロックを配置する。ブロックノ寸法は、内部の配置を変えることにより幾つかの候補のうちから選べる。ブロックの間は配線領域で、水平チャネル・垂直チャネルに分割する。相対配置は、各ブロックがどのチャネルを介して隣あっているかで示す。さらにこの隣合うブロック同志の間をどれだけ開ければ、配線ができるかを推定し、その距離をブロック寸法最適化の条件として与える。この推定処理はまた、ブロックの絶対配置を決めてチップの大きさを見積もる役割を持っているので、チップ面積推定処理とよんでいる。

2．2　ブロック寸法最適化処理の概要

ブロック寸法最適化処理は、相対配置，配線領域幅をそのままにし、チップ縦横比を指定範囲に納める条件のもとで、チップ面積が最小となる各ブロックの寸法の組み合わせを求める処理である。

図２に従来のブロック寸法最適化処理の概要を示す。各処理の手順を以下に示す

（１）制約行列生成

ブロックの寸法選択条件、ブロックの隣接条件、チップの縦横比条件を表す制約式を作る。変数は、ブロックの寸法候補選択変数 $\lambda_i^{(j)}$（ブロックｉの，ｊ番の候補を選択するかどうかを０－１で表す）、ブロックの原点座標 x_i, y_i、チップの寸法 x_c, y_c である。ブロックｉの寸法候補を$(w_i^{(j)}, h_i^{(j)})$ブロックｉ，ｊ間の配線領域幅を$(\varepsilon_{ij}, \delta_{ij})$，チップの縦横比の許容範囲を$(r^-, r^+)$とすると、条

件はそれぞれ次のように表される。

最小化：　$\sqrt{r^- r^+} \cdot x_c + y_c$　　(2-1)

条件①：（ブロック寸法選択条件）

$$w_i = \sum_{j=1}^{Ni} \lambda_i^{(j)} w_i^{(j)}, \quad h_i = \sum_{j=1}^{Ni} \lambda_i^{(j)} h_i^{(j)},$$

$$\sum_{j=1}^{Ni} \lambda_i^{(j)} = 1, \quad 0 \leq \lambda_i^{(j)} \leq 1 \quad (j = 1, \cdots, Ni). \qquad (2-2)$$

条件②：（ブロック隣接条件）

$$x_j - (x_i + w_i) \geq \delta_{ij}$$
$$y_j - (y_i + h_i) \geq \epsilon_{ij} \qquad (2-3)$$

条件③：（チップ縦横比条件）

$$r^- \leq y_c / x_c \leq r^+ \qquad (2-4)$$

以上の式から、係数行列すなわち、ＬＰ制約行列を生成する。

（２）ブロック寸法選択

制約行列で表わされる線形計画問題（ＬＰ）をシンプレックス法を用いて解く。各ブロックに対し、ＬＰの解として得られたブロック寸法選択変数の値に一番近い寸法を候補の中から選ぶ。

（３）相対配置の微小変更

選択されたブロック寸法をもとに詰め合わせを行い絶対座標を求める。相対配置をこれに適合するように少し変更する。

相対配置を変更すると、配線領域幅が多少変化する可能性があるためより正確を期するため、ブロック寸法最適化処理のあとでチップ面積推定を行う。

２．３　従来の処理の問題点

従来のブロック寸法決定処理では、問題を解くため線形計画法を使用しているので、ブロック数の２乗に比例するメモリと少なくとも３乗に比例する処理時間が必要であり、大規模化するＶＬＳＩに対処することが困難であった。

３．　ブロック寸法最適化処理の階層化

３．１　基本方針

メモリ量，処理時間の削減のためには、いちどに扱うブロック数を少なくすればよい。領域をいくつかに分割し、それぞれの領域でブロック寸法最適化を行えば、これが実現できる。しかし、それだけでは、領域の大きさが合わなくなるため、全体としての最適化は行えない。これを解決する有効な手段として、処理の階層化を考えた。まず，ブロックの寸法を変えることにより、各領域の寸法候補を幾つか用意する。次に、それぞれの領域をブロックとみなして、全体でブロック寸法の最適化を行う。最後に、決まった領域の寸法に対応するブロックの寸法を決めればよい（図３）。

原理的には、各領域の寸法の候補としてすべてのブロックの寸法の組み合わせを尽くして得られるもの総てを考えることになる。しかし、ある寸法Ａ

が縦横ともある寸法Bより大きかったら、Aが採用される可能性はない。すなわち、ある縦横比の範囲で局所的に面積最小の寸法だけを考えれば十分である。この寸法は、従来のブロック寸法最適化処理を応用すれば得られる。

ブロック数が非常に多い場合には、分割した領域をさらに分割して、階層化すればよい。

3.2 領域の階層分割

処理を階層化するには、まず領域を階層的に分割する必要がある。以下では、一つの領域を最大の部分矩形領域に分割する手順を示す。

相対配置は、チャネルの接続関係で表わされているから、領域の分割もこれをもとに行う。矩形領域は、それを囲む4つのチャネルで表わされる。領域を縦に分割する場合は、左端と右端との両方につながるチャネルを捜してそこで分割すればよい。横方向の分割も同様である。

以下に四畳半型配置の分割の手順を示す（図4参照）。

STEP 1: 領域の右上にある、できるだけ大きい矩形の部分領域を捜す。

STEP 2: 見付かった領域を除去する。

STEP 3: 領域が残っていない場合は終了。領域がまだある場合は、できるだけ右上にあるできるだけ大きい矩形の領域を捜し、STEP 2 に戻る。

3.3 階層的ブロック寸法最適化処理

階層的ブロック寸法最適化処理は図5-(a)に示す様に①領域寸法候補作成処理、及び②領域寸法選択処理の二つからなる。

①は、各領域に対し、その部分領域の寸法候補、及び相対配置条件をもとに局所最小な複数個の寸法候補を作成する処理である。図5-(b)に領域寸法候補作成処理の手順を示す。ＬＰを解く際に制約条件として領域の縦横比を縦長から横長に変えることによって、複数の局所最小な寸法を得ている。一つの寸法候補が得られたら、そのときの部分領域の寸法を対応させて記録しておく。

②はチップ面積を最小化するブロックの寸法を選択する処理である。図5-(c)に領域寸法選択処理の手順を示す。まずチップの寸法候補の中から、面積最小な寸法を選ぶ。以下、ある領域の寸法候補が選ばれたら各部分領域対応するの寸法を選択する。これをすべてのブロックの寸法が決まるまで繰り返す。

3.4 使用メモリ量と処理時間の理論式

ブロック数をN、部分領域数をM、領域当たりの寸法の上限をL、ブロックあたりの寸法の候補数をKとする。議論を簡略化するため、配置階層木が完全M分木になっているとする。ブロックだけからなる領域のレベルを0、階層を上がるごとに、領域のレベルを1増やすものとし、最大のレベルをJとする。Mは2から$\sqrt{N}$までの値を取りうる。

（1）使用メモリ量

使用メモリ量は、主に制約行列及び、領域寸法候補テーブルによって決まる。制約行列の大きさは、Mの2乗とLに比例する。

領域の寸法の個数は配置階層木で、1レベル上にいくごとにおよそM倍に

なる。従って、領域寸法候補テーブルの大きさは、レベルｉの部分領域一つに対して min(Ｌ，ＫＭ)Ｍ だけ必要である。レベルｉの部分領域は、Ｎ／Ｍ 個あるから、全体で最大 ＫＭＮlogＮ／logＭ 必要となる。

Ｍを定数とすれば、制約行列の大きさは一定であり、寸法候補テーブルは ＮlogＮ に比例する。Ｍ＝ $\sqrt{N}$ とすると、それぞれＮ，Ｎの１．５乗に比例する。一括処理の場合は、制約行列に ＫＮ に比例するメモリが必要であるから、メモリ量の削減効果は十分ある。

（２）処理時間

ＬＰ制約行列の大きさをｍ×ｎとするとき、一回当たりのＬＰの計算時間が $m^{\alpha} \cdot n^{\beta}$ に比例するとする。階層的に処理した場合、レベルｉでは制約行列の大きさは、３Ｍ×（min(Ｌ，KM^{i})＋２Ｍ）となる。以下の議論では計算量のオーダーのみを考えるので、定数および微小な項を無視する。

$$T \propto \sum_{i=1}^{J} M^{\alpha} (KM^{i})^{\beta} KM^{i} N/M^{i} + \sum_{i=J+1}^{n} M^{\alpha} (LM)^{\beta} LN/M^{i}$$

$$= M^{\alpha} K^{\beta+1} N \sum_{i=1}^{J} M^{i} + M^{\alpha+\beta} L^{\beta+1} \sum_{i=J+1}^{n} M^{-i}$$

$L = KM^{J}$ を使い主要項を残せば

$$\cong KL^{\beta} M^{\alpha} N (1 + M^{\beta-1}) = KL^{\beta} M^{\alpha+\beta-1} N \qquad (3-1)$$

が得られる。

Ｍを定数に取れば、$T \propto KL^{\beta}N$ である。$\alpha = 2$，$\beta = 1$，Ｍ＝ $\sqrt{N}$ とすれば、$T \propto KLN^{2}$ となる。一括の場合Ｎの３乗以上かかる。これと比べ処理時間の削減効果が期待できる。

４．評価結果

（１）評価対象品種

２０ＫゲートＶＬＳＩを用いて評価を行った。一つのＲＡＭと４０個のランダム論理ブロックを含む。ランダムブロックの寸法候補を各７種類与えた。すなわちＮ＝４１，Ｋ＝７である。

（２）評価パラメータ

評価のパラメータは次の２つである。

①部分領域数上限：Ｍ

階層分割をするときに１領域を最大何個まで分割するかを示す。

②領域当り寸法候補数上限：Ｌ

分割した領域が最大何種類までの寸法の候補を取りうるかを示す。

（３）評価項目次の３点について評価を行った。

①処理時間

ブロック寸法最適化に要した時間

②メモリ使用量

ブロック寸法最適化に要したメモリ量の最大値

③チップ面積

ブロック寸法最適化後のチップ面積及びそれに対するチップ面積推定値。

4．1　評価結果

（２）メモリ使用量

図６に、部分領域数ｍの上限とメモリ使用量の関係を示す。評価の範囲ではブロック寸法候補テーブルの大きさは小さくて済むので、ＬＰ制約行列が占めるメモリ量を示した。図６より、メモリ使用量Ｓは、

$$S \propto M^2 L$$

で与えられる。これは理論の示すところと一致している。

（２）処理時間

図７に処理時間と部分領域数の関係を示す。この図から、処理時間ＴはＭの大きいところでは、

$$T \propto M^{1.5} L^{0.7}$$

で与えられる。これは、式（３－１）で$\alpha = 1.8$、$\beta = 0.7$とした場合に相当する。Ｌ，Ｍが大きい場合、一括での処理より処理時間がかかっているが、これはＬＰを解く回数が一括の場合１回だけなのに対し、階層化した場合かなりの回数解く必要があり、そのオーバーヘッドの為である。Ｎが大きくなればこの効果が相対的に小さくなるので、処理時間を減少させられる。

（３）チップ面積

図８に、部分領域数の上限とチップ面積の関係を示す。傾向として、部分領域数を多くして、階層を少なくしたほうが、チップ面積が小さい。部分領域が少ないと、チップ面積が大きくなり実用的でない。しかし部分領域数を１６程度で寸法候補数を十分多く取れば、一括の場合との差は面積的に１０３％以下であって、十分実用に耐えると考える。

5．結　言

ブロック寸法最適化処理の高速化、メモリ使用量削減のため処理の階層化を行った。メモリ使用量をブロック数Ｎの２乗から最大Ｎの１．５乗のオーダーに、処理時間をＮの３乗から最大Ｎの２乗のオーダーに削減した。２０ＫゲートＶＬＳＩによる評価を行い、一括処理に比べてメモリを１／６，処理時間を２／３に削減し、チップ面積は１０３％に押さえることができた

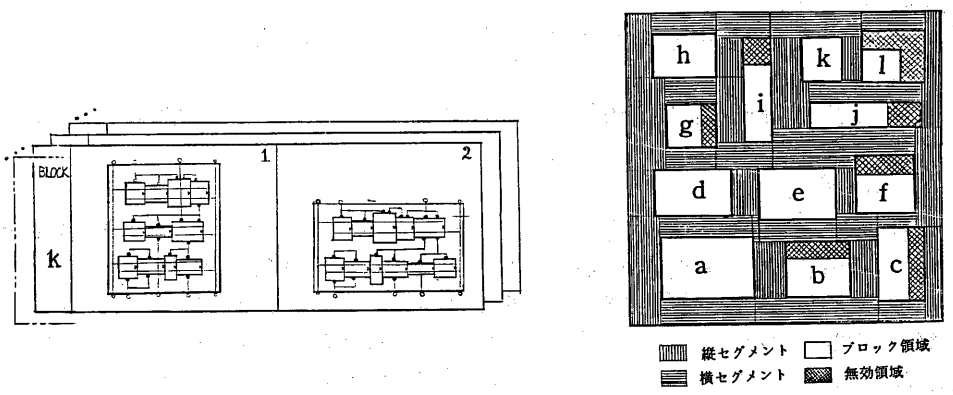
参　考　文　献

(1) H. Terai, et. al.; "Automatic Placement and Routing Program for Logic VLSI Design Based on Hierarchial Layout Method," Proc. of Conf ICCC, September 1982.

(2) T. Kozawa, et. al.; "Combine and Top Down Block Placement Algorithm for Hierarchical Logic VLSI Layout," Proc. of 21st D. A. Conf., June 1984, pp. 667-669.

(3) 茂垣他：線形計画法を用いたブロック寸法最適化配置手法，CAS-84-119, pp29-35

(4) Otten, R. H. J. M: "Efficient Floorplan Optimization," ICCD Nov. , 1983, pp. 499-502.



(a) ブロック　　(b) チップ

図1. レイアウトモデル

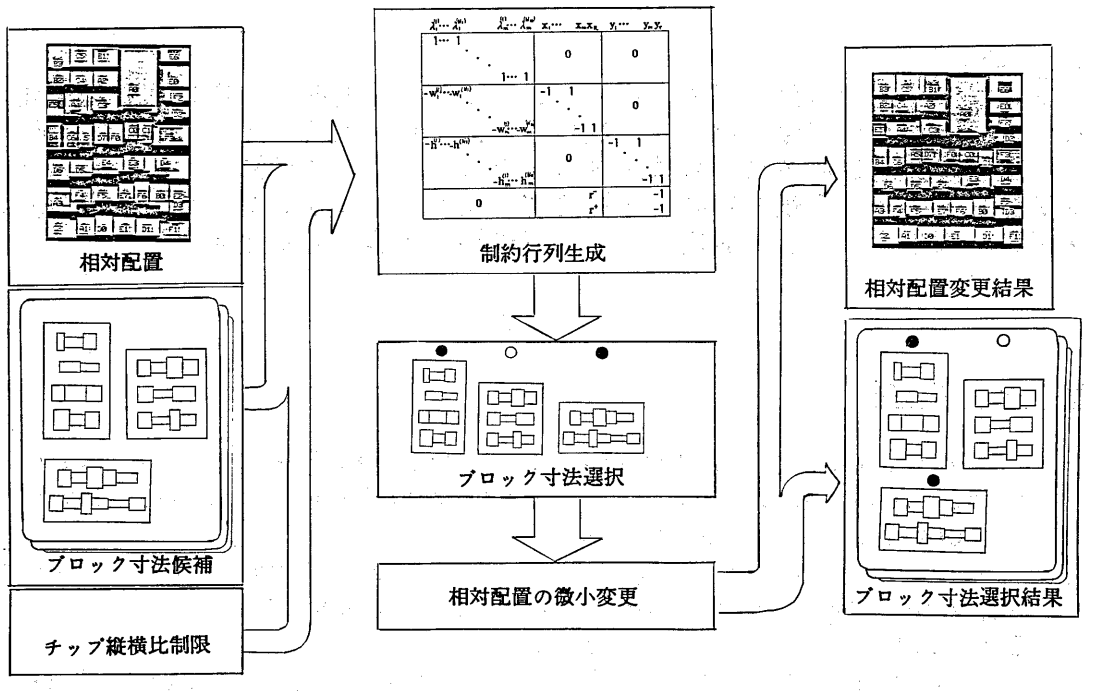


図2. 従来のブロック寸法最適化

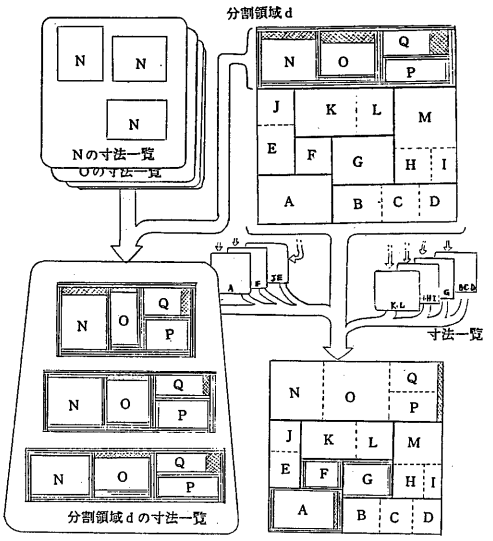


図3. 階層的ブロック寸法最適化の考え方

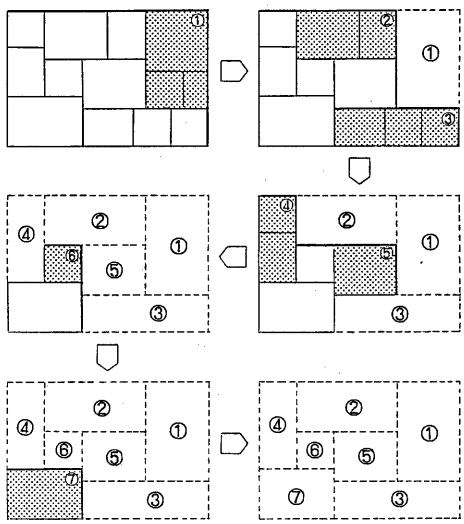
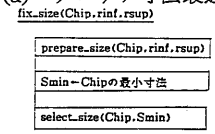
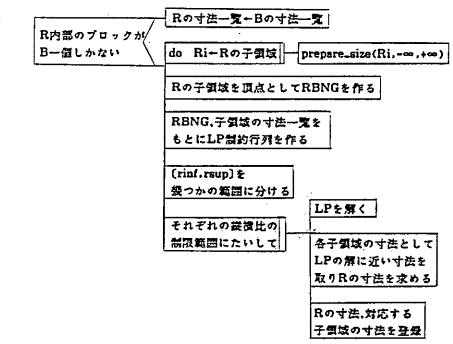


図4. 領域の分割手順

(a) ブロック寸法最適化



(b) 領域寸法候補作成



(c) 領域寸法選択

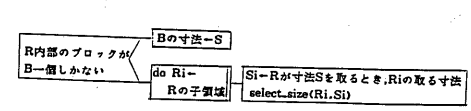


図5. 階層的ブロック寸法最適化処理

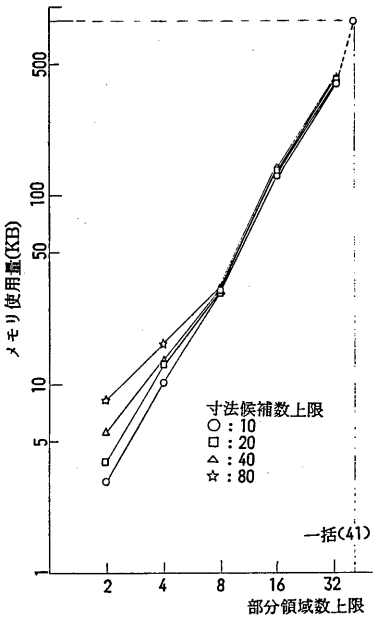


図6. メモリ使用量

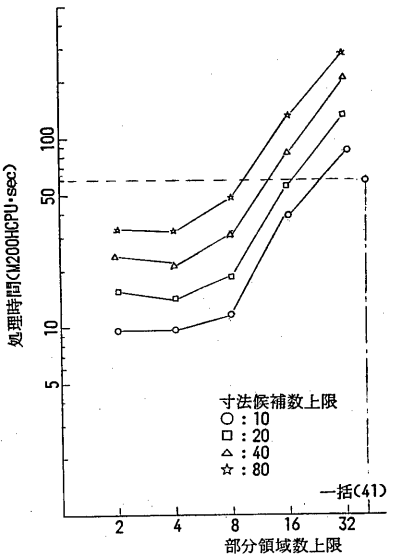


図7. 処理時間

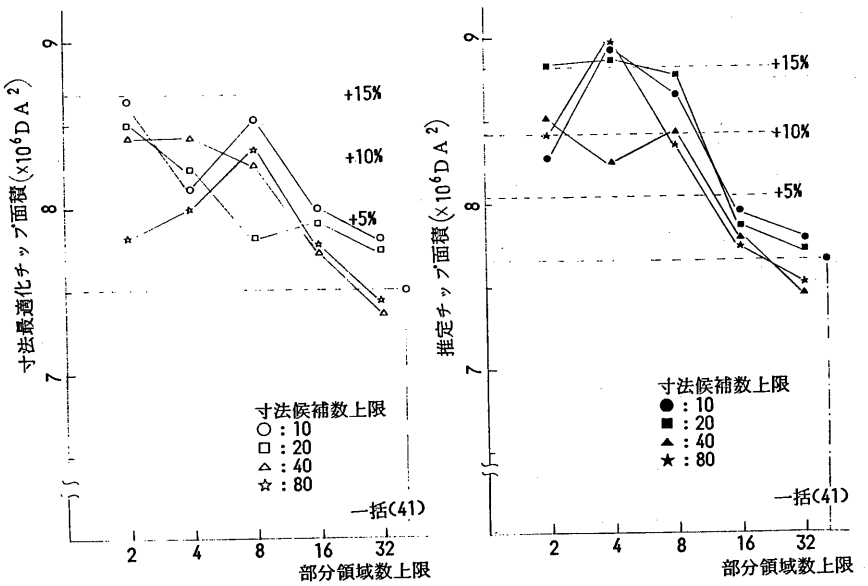


図8. チップ面積