

任意形状ブロックを対象とした一配置手法

中谷 直司*, 金杉 昭徳*, 近藤 邦雄*

A Placement Method for Arbitrary Shape Blocks

Naoshi NAKAYA*, Akinori KANASUGI* and Kunio KONDO*

*埼玉大学工学部 (〒338-8570 埼玉県浦和市下大久保255)

*Faculty of Engineering, Saitama University (255 Shimo-okubo, Urawa-shi, Saitama 338-8570)

概要 VLSIレイアウト手法の多くにおいては、配置対象ブロックの形状は矩形に限定され、矩形以外のブロックを扱うことは困難である。また矩形以外のブロックを対象とする手法においても、配置対象ブロックの形状自由度は高くない。そこで本論文では、従来手法では難しかった凹部を含むブロックの配置や、配置領域を制限しての配置を可能にする、任意形状ブロックの配置手法を提案する。提案手法は、ブロックの形状に応じた評価値を定義し、その値を用いることで任意形状ブロックを配置する組み立て式配置手法と、その配置順序に遺伝的アルゴリズムによる制御を組み合わせたものである。また最後に、計算機実験により本手法の有用性を示す。

Abstract

In this paper, we propose a new placement method, which can treat of arbitrary shape blocks. In the proposed method, we place the blocks using the value that is based on the shape of block. The method is able to handle the blocks that include notches with restricted layout area. We can also search effective placements because the order of placement is controlled by Genetic Algorithm. We have made computer experiments by the automatic placement tool for proposed method. The computer experiments show that the results of proposed methods are effective placements without wasteful area.

Key Words: CAD, Layout, Placement, Arbitrary Shape Block, Genetic Algorithm

1. まえがき

多くのVLSIレイアウト手法¹⁾²⁾においては、配置対象ブロックの形状は矩形に限定され、矩形以外のブロックを扱うことは困難である。また矩形以外のブロックを対象とする手法においても、そのブロック形状はL型³⁾や凸XY多角形と呼ばれる形状⁴⁾などに限定され、配置対象ブロックの形状自由度は決して高くなく、凹部を含むブロックの配置などには対応できない。また、ブロック形状に制限がない手法⁵⁾も提案されている。しかし、文献4)の手法も含め、コンパクション手法であるため、配置領域を限定して配置を行うことは困難という問題がある。また、これらコンパクション手法を用いるには、何らかの初期配置が必要になるが、任意形状ブロックに適した初期配置手法は提案されていない。

そこで本論文ではこれらの問題に対し、形状に応じた評価値を用いることで凹部を含むブロックの配置や、配置領域を制限しての配置も可能とする任意形状ブロックの配置手法を提案する⁶⁾。本手法はVLSIレイアウトにおける組み立て式配置手法の一種であるが、その配置順序に遺伝的アルゴリズム⁷⁾⁸⁾による制御を組み合わせることで、組み立て

式手法に共通する欠点である配置結果の順序依存問題にも対応可能である。また、本手法はすでに提案されているコンパクション手法の初期配置などにも有効である。

本手法の有用性を確認するため計算機実験を行った結果、凹部を含む完全な任意形状ブロックの配置や、配置領域に制限を設けての配置も可能なことが確認された。

2. 任意形状ブロックの組み立て式配置手法

本節ではブロック形状に応じた評価値を定義し、その値を用いて任意形状ブロックを配置する組み立て式配置手法を提案する。本手法では配置領域に制限を設けての配置を、配置領域に制限がないときの特殊な場合としてとらえ、配置を行う。したがって、ここでは基本となる配置領域に制限がない場合の提案手法から述べる。なお、本手法では組み立て式手法において重要な配置順序の制御を、遺伝的アルゴリズムにより行い、良好な配置結果を得ているが、遺伝的アルゴリズムによる制御に関しては後述する。

2.1 配置領域に制限がない場合の手法

配置領域に制限がない場合の任意形状ブロックの配置について考える。ここでは初めに、任意形状ブロックとブロック形状に応じた評価値を定義する。次に、定義した評価

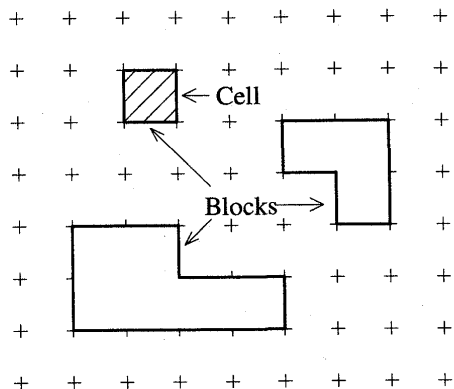


図1. 任意形状ブロックの例

値を使いブロックを配置する手順について述べ、さらに仮想配線長を用いて配置手順の拡張を行う。

2.1.1 任意形状ブロックと隣接度

本手法で対象とする任意形状ブロックとは、図1に示すようないくつかのセル（1つのセルは図に斜線で示したもの）の塊である。また、レイアウト領域は基盤の目のように正方形の区画が縦横に連なったものとし、1つのセルは1つの区画を占めるとする。

本手法では、レイアウト領域の各区画に対してブロック形状に応じた評価値を定める。この評価値は、各区画の周りの8つの区画にいくつセルがあるかを数えたものであり、その区画にセルを置いたとき、すでに配置されているセルと隣接する割合を示している。したがって、以下ではこの評価値を「隣接度」とよぶこととする。すなわち座標 (x, y) の区画の隣接度 R_{xy} は次式で与えられる。

$$R_{xy} = \sum_{i=-1}^1 \sum_{j=-1}^1 S_{x+i, y+j} \quad (1)$$

ただし S_{xy} : 座標 x, y の区画にセルが存在するとき 1
 〃 しないとき 0

なお、隣接度はすでにセルが配置されている区画については定義しない。

例として図2の場合、中央の区画（座標 (x, y) ）の隣接度は式(1)において、座標 $(x, y+1)$ と $(x+1, y)$ と $(x+1, y+1)$ の区画にセルが存在するので、

$$R_{xy} = 0+0+0+0+0+1+0+1+1 = 3$$

となり、隣接度は3である。同様にすべての区画の隣接度

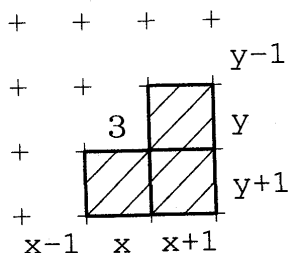


図2. 隣接度の例

を求めることができる。

2.1.2 隣接度による配置

前述したように隣接度はその区画にセルを配置したとき、すでに配置されているセルと隣接する割合を示している。そこで、ブロックを配置するときは配置した区画の隣接度の和が最大になる区画に配置すると効率がよく、チップ面積の最小化にもつながる。したがって、ブロック配置の手順は以下ようになる。

- ・ステップ1：レイアウト領域の各区画に対して式(1)に従い、隣接度を求める。
- ・ステップ2：ブロックを配置したとき、隣接度の和が最大になる区画を選ぶ。なお、このときすでにセルの存在する区画を選ぶことは許されない。
- ・ステップ3：ステップ2で選んだ区画にブロックを配置する。
- ・ステップ4：配置するブロックがなくなるまでステップ1, 2, 3を繰り返す。

例として図3-aのようにセルがすでに配置されているときに、図3-bのブロックを配置する場合を考える。ステップ1に従い隣接度を求めると、その分布は図3-cのようになる（図に示した以外の区画の隣接度は0である）。この隣接度分布から考えてステップ2で選ばれる区画は、隣接度の和が8で図3-dの斜線の部分になる。したがって、ステップ3で配置した結果は図3-eとなる。ここでまだ配置するブロックが残っている場合は、図3-eに示したセル配

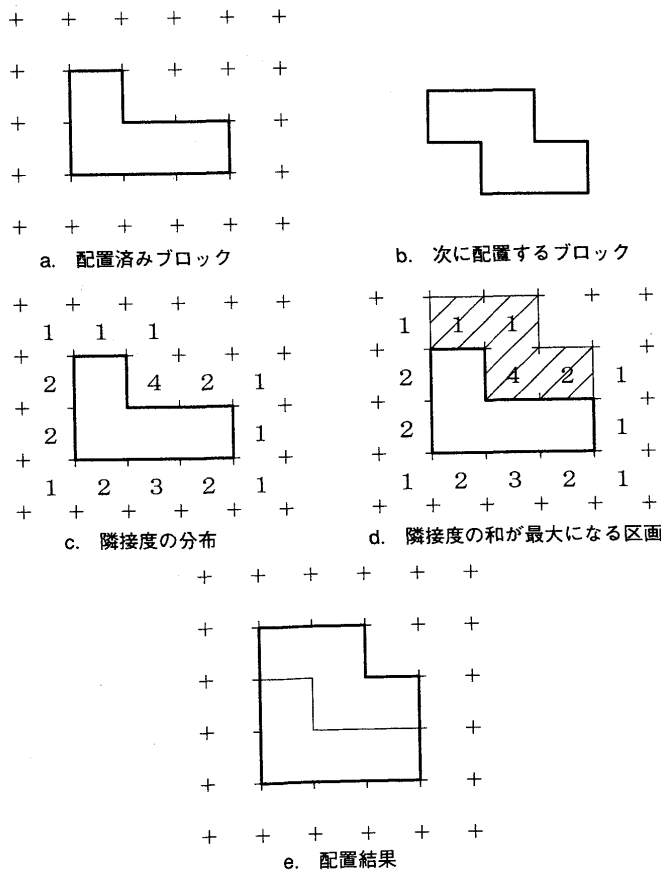


図3. 隣接度による配置の例

置に対し隣接度分布を求め(つまり, ステップ1に戻る), 同様に繰り返す。以上が今回提案する組み立て式配置手法の基本である。

なお, すでに述べたように, 本手法は配置対象である任意形状ブロックを, 正方形のセルが縦横に連なったものとしている。したがって, ブロック各辺の長さはセルの長さの整数倍に制限され, この点での自由度は決して高いとはいえない。もちろんセルを十分に小さく考えれば, 理論的にはどのような精度でもブロックを表現できるが, その場合, 精度に応じた処理時間の増大を避けることはできない。そこで, 長さについて高い精度を必要とするときは実用性を考慮して, 次のような方法で本手法を用いることを提案する。まず, ブロックの各辺の長さを実用上問題のない程度に近似し, 本手法を用いて配置する。その上で, 近似したブロックを本来のブロックに置き換える。すると, ブロック同士の重なりや, あるいは無駄領域を生じることになるが, それらを解消するコンパクション手法⁵⁾はすでに提案されている。そこで, この従来手法を用いてコンパクションをかける。つまり, 本手法を使って従来のコンパクション手法の初期配置を与えることで, 長さについて十分な精度を持ち, なおかつ実用的な処理時間での配置が可能となる。

2.1.3 仮想配線長による拡張

隣接度は形状に応じた評価値であるため, 形状が対称の場合, 隣接度の分布も対称となり, ブロックを配置したときに隣接度の和が最大になる区画も複数になる。そこで配置区画候補が複数になったときの選択法として, 他の評価法との併用が必要となる。本手法では仮想配線長を用いることにし, これによりブロック配置の手順は次のように拡張される。

- ・ステップ1: レイアウト領域の各区画に対して式(1)に従い隣接度を求める。
- ・ステップ2: ブロックを配置する区画を選ぶ。なお, このとき, すでにセルの存在する区画を選ぶことは許されない。
- ・ステップ2.1: ブロックを配置したとき, 隣接度の和が最大になる区画を選ぶ。
- ・ステップ2.2: ステップ2.1で選ばれた区画が複数存在するときは, その中から配置したとき仮想配線長が最短になる区画を選ぶ。
- ・ステップ2.3: ステップ2.2で選ばれた区画も複数存在するときは, それらの区画は等価として, 探索時に先に探索された区画を選ぶ。
- ・ステップ3: ステップ2で選んだ区画にブロックを配置する。
- ・ステップ4: 配置するブロックがなくなるまでステップ1, 2, 3を繰り返す。

ステップ2.2での仮想配線長は, 各ネットごとにそれ

を構成する端子を内包する最小矩形を考え, その周囲長の1/2をそのネットの仮想配線長とするものを採用する。なお本手法の場合, 配置過程で仮想配線長を算出する必要はあるため, ネットを構成するすべての端子が配置済みとは限らない。そこで, 配置過程では配置されていない端子は無視し, 配置済みの端子のみで仮想配線長を算出する。すなわち, ネット n_i に含まれる配置済み端子のリストを $P_i=(p_j \cdots p_k)$ とし, それらの端子の x 座標を $p_{j,x}$, y 座標を $p_{j,y}$ とするとネット n_i の仮想配線長 $n_{i,length}$ は,

$$n_{i,length} = \max_{P_i} (|p_{j,x} - p_{k,x}|) + \max_{P_i} (|p_{j,y} - p_{k,y}|)$$

となり, したがって総ネット数を C とすると, 総仮想配線長 D は,

$$D = \sum_{i=1}^C n_{i,length}$$

となる。また, 上記手順ではステップ2.3が定められており, 探索順という明確な根拠のない選択がなされている。しかし, 本手法では仮想配線長の算出が端子位置に基づいているため, 実際的なブロックではステップ2.2で仮想配線長が最短になる区画が複数存在することは, きわめてまれになる。なお, 本論文で後述する計算機実験の最終配置結果においても, ステップ2.3の利用はなされていない。ただし, 配置済みブロックと配置しようとしているブロックの間にネットが存在しない場合にはステップ2.2での特定ができないため, ステップ2.3が適用される。しかし, こういった場合は, 複数ある配置区画候補のいずれに配置しても等価であるとして, 本手法では探索順による配置を採用した。

2.2 配置領域に制限がある場合の手法

次に, 配置領域に制限がある場合について考える。すでに述べた配置領域に制限がない場合においては, レイアウト領域は無限の広がりを持つものとして扱ってきた。そこで配置領域に制限がある場合においても同様に, レイアウト領域は無限の広がりを持つものとする。そして配置領域制限は, レイアウト領域の配置可能区画以外はすべてセルで埋めつくされている, とすることで実現する。これにより, 配置領域に制限がある場合においても同様に隣接度を求め, その和が最大になる区画に配置するという手法を用いることができる。ただし, この場合, 配置可能区画に制限が存在するため, 配置対象となるブロックがすべて配置可能とは限らない。そこで, 配置手順は次のように修正される。

- ・ステップ1: レイアウト領域の各区画に対して式(1)に従い隣接度を求める。
- ・ステップ2: ブロックを配置する区画を選ぶ。なお, このときすでにセルの存在する区画を選ぶことは許されない。また, すでに十分な配置可能区画が存在しないときは, 未配置のブロックがあっても配置手

順を終了する。

- ・ステップ2.1：ブロックを配置したとき、隣接度の和が最大になる区画を選ぶ。
- ・ステップ2.2：ステップ2.1で選ばれた区画が複数存在するときは、その中から配置したとき仮想配線長が最短になる区画を選ぶ。
- ・ステップ2.3：ステップ2.2で選ばれた区画も複数存在するときは、それらの区画は等価として、探索時に先に探索された区画を選ぶ。
- ・ステップ3：ステップ2で選んだ区画にブロックを配置する。
- ・ステップ4：配置するブロックがなくなるまでステップ1, 2, 3を繰り返す。

例として、4行5列の配置領域制限がある場合について述べる。このときレイアウト領域は図4-aのように4×5の配置可能区画を残してセルで埋めつくされていると考える（図に示した以外の区画は、すべてセルで埋めつくされている）。これに対し隣接度を式(1)に従い求めると、図4-bのような分布が得られる（図に示した以外の区画の隣接度は0である）。後は配置領域に制限がない場合と同様に、すでに述べたブロック配置の手順に従いブロックを配置する。なお、この例では配置領域制限を矩形としたが、本手法としては領域制限が矩形である必要はない。たとえ領域制限が非矩形であっても、隣接度を求め配置することは可能である。

また、配置領域制限内に配置対象のブロックが収まるかどうかは重要な問題ではあるが、今のところそれを事前を知る方法は知られていない。もちろん、領域制限の面積よりもブロックの総面積が大きい、言い換えるならば配置可能区画の数よりも、ブロックの総セル数が多い場合は収まらないのは明らかである。しかし、面積の大小関係に問題がないからといって必ずしも収まるとは限らない。本手法において配置領域制限内に配置対象のブロックが収まるかどうかは、一般的な「ポリオミノの敷き詰め」問題と同様な問題といえるが、この問題において可能を証明する手段がないように、本手法も収まることを証明する手段はない。ポリオミノの敷き詰めと同様に、パリティを使った方法で不可能を証明⁹⁾することはできるが、それは可能を証明したことにはならない。すなわち、収まらないと証明されな

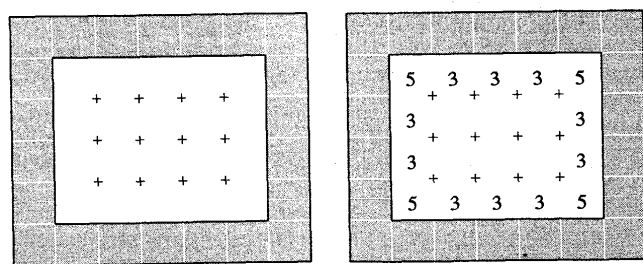


図4. 領域制限があるときのレイアウト領域

かったとしても収まるとは限らない。あくまでも収まる可能性があるだけであり、実際に収まるかどうかは別の問題である。以上のような理由から、配置領域制限内に全ブロックが収まるかどうかを事前に知ることは現状では困難である。

3. 遺伝的アルゴリズムによる配置順序制御

本手法では配置結果は配置順序に依存する。これは本手法が組み立て式手法の一種であるため、配置処理過程では全体的な見通しがなされないことに関係する。すなわち、図5のようなA, B, Cの3つのブロックを配置するとき、A, B, Cの順で配置すると図5-aのような結果が得られ、A, C, Bの順で配置すると図5-bのような結果になる。この場合、最終的結果としては図5-aの配置結果が形状としては優れているが、これは手法としてこの配置を目指していたわけではなく、配置順序から偶然に得られた結果にすぎない。すなわち配置結果は配置順序に強く依存するため、本手法では配置順序に適切な制御を必要とする。そこで本研究では遺伝的アルゴリズム (GA) を用いて、本手法に最適な配置順序を求めることとする。GAは生物が遺伝と進化により環境に適應することに範をとった最適解探索アルゴリズムの一つであり、広範囲にわたる探索に威力を発揮することで知られている。しかし、GAを用いて順列組み合わせ問題を解く場合には、コーディングやオペレータに何らかの工夫をしなければ満足のいく解は得られない¹⁰⁾。そこで本手法では、次のようなコーディングの工夫により問題を解決する。

ブロックの配置順序を $T=(t_1 \cdots t_n)$ とし、またブロックを適当な順序で並べたリスト W (定数) を用意する。そして、 i 番目に配置するブロックを、未配置ブロックリスト $W - \{t_1, \dots, t_{i-1}\}$ の何番目であるかを表し、これを遺伝子 l_i とし、リスト $L=(l_1 \cdots l_n)$ を染色体とする。このコーディングにより、遺伝子型と表現型が一对一に対応づけられ、通常のオペレータの適用が可能になる。なお、このコーディングは Grefenstette らによる巡回セールスマン問題のための方法¹¹⁾ を、ブロックの配置順序に応用したものである。

例としてブロックの配置順序が $T_1=(B D A C E)$ のときのコーディングを考える。ブロックを適当な順序で並べたリストを $W=(A B C D E)$ とすると、1番目に配置するブロックBは、未配置ブロックリスト W の2番目なので遺伝子 l_1 は2となる。次に2番目に配置するブロックDは、未

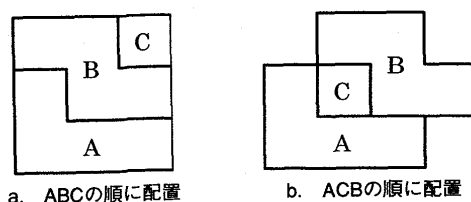


図5. 配置順序依存の例

配置ブロックリスト $W-\{B\}=(A C D E)$ の 3 番目なので遺伝子 l_3 は 3 となる。以下これを繰り返すと染色体はリスト $L_1=(2 3 1 1 1)$ として求まる。また、ブロックの配置順序が $T_2=(E D C B A)$ のときの染色体は、リスト $L_2=(5 4 3 2 1)$ となる。GAでは、これらの染色体を交叉することで新しい染色体を作り探索を行うが、ここでは上記のコーディングの工夫により、通常の交叉を用いての探索が可能になる。すなわち、上記の2つの染色体の例で考えると染色体はそれぞれ、

$$L_1=(2 3 1 1 1)$$

$$L_2=(5 4 3 2 1)$$

である。これを 2 番目の遺伝子と 3 番目の遺伝子の間で交叉したとすると、新しい染色体は、

$$L'_1=(2 \underline{3} 3 2 1)$$

$$L'_2=(5 4 \underline{1} 1 1)$$

となる。これをコーディングと逆の手順でブロックの配置順序に戻すと、

$$T'_1=(B D E C A)$$

$$T'_2=(E D A B C)$$

となり、ブロックの重複を含まない配置順序、すなわち順列という条件を満足する解が得られ、効率の良い探索が可能になる。なお、突然変異は遺伝子 l_i を、条件 $1 \leq l_i \leq N-i+1$ を満足する遺伝子にランダムに置き換えることで実現する。

GAを用いる上での評価値である適応度 f は、仮想配線長 D と総ネット数 C 、さらに配置結果を内包する最小矩形を考えたときに、セルが配置されていない区画の数 E から

$$f = \frac{1}{1 + \alpha \frac{D}{C} + \beta E} \quad (2)$$

として定義する。すなわち、仮想配線長が短く、かつ配置結果が隙間なくコンパクトにまとまっている方が評価が高いとする。また、配置領域に制限がある場合には、最終的に未配置のブロックが存在することが考えられるが、この定義を用いればセルが配置されていない区画の数 E が増えるため評価が低くなり、世代を重ねることで淘汰されると考えられる。なお、 α と β は、仮想配線長と配置結果の隙間という二つの評価要素の、適応度 f に占める割合を決定する係数であり、配置の目的に応じて設定する。

GAによる配置順序制御の導入により、本配置手法全体は次のようになる。

- ・ステップ 1：初期集団として M 個の個体を生成する。
このとき各個体の持つ情報は、上記のコーディングによりブロックの配置順序を表している。
- ・ステップ 2：各個体に対し次の処理を行う。
- ・ステップ 2.1：個体の持つ配置順序情報に従い、前述のブロック配置の手順を使って配置可能なすべてのブロックを配置する。このとき配置結果は配置順

序により、一意に決定される。

- ・ステップ 2.2：配置結果から仮想配線長 D と空き区画数 E を求め、式(2)に従い適応度 f を算出する。
- ・ステップ 3：ステップ 2 で求めた適応度に従い、選択、交叉、突然変異などのオペレータにより個体集団を進化させる。
- ・ステップ 4：ステップ 2, 3 を一世代とし定められた世代数を繰り返す。
- ・ステップ 5：最終的に得られた最も優れた適応度を持つ個体の配置結果を、本手法の配置結果とする。

4. 計算機実験

本手法の有用性を確認するため、計算機実験を行った。遺伝的アルゴリズムのオペレータとパラメータは、経験的に次のように決定した。これは以下の実験において共通である。

- ・適応度：適応度 f は式(2)において、仮想配線長による評価と配置結果の隙間による評価を適度に取り入れることを目的に、 $\alpha=\beta=1$ とする。
- ・選択：適応度比例戦略とエリート保存戦略を用いる。
なお、このときのエリート保存数は 1 とする。
- ・交叉：一点交叉を用いて、交叉確率 p_c は 0.95 とする。
- ・突然変異：突然変異確率 p_m は 0.1 とする。

まず、図 6 のブロックを配置する実験例を示す。このとき個対数 M を 30、世代数 T を 20 とした。配置領域に制限を設けないときの配置結果を図 7 に示す。なお、配置順序はブロック番号 5, 3, 2, 6, 4, 1, 0 の順で、仮想配線長はセ

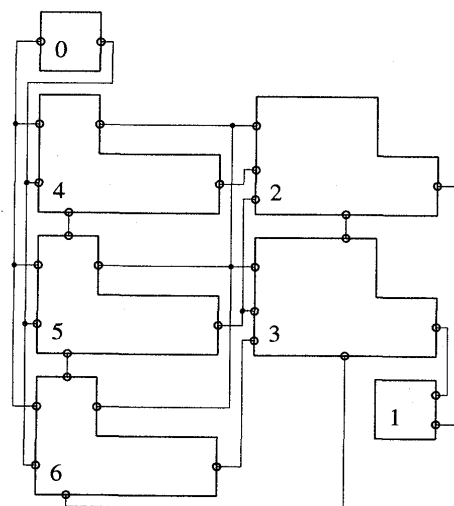


図6. 配置対象ブロックの例 1

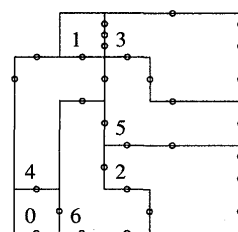


図7. 領域制限がないときの配置 (例 1)

ルの一辺を1として46.5である。配置結果を内包する最小矩形を考えると5×5の区画になり、その中で左上角の区画が空いているので空き区画数は1となる。また、配置領域に制限がある場合についても計算機実験を行った。ここでは例として図6に示したブロックを配置することを考える。図6に示したブロックの総セル数は24、そこで4行6列の配置領域制限を設けることとした。結果は図8に示すように、与えられた配置領域内にすべてのブロックが配置された。さらに本手法においては、配置領域制限は必ずしも矩形である必要はない。そこで図6に示すブロックを図9のような非矩形の配置領域へと配置した。結果は図10に示すように、与えられた配置領域内にすべてのブロックを配置することができた。

も行い、図12に示したようにすべてのブロックが配置されることを確認した。

5. むすび

本論文ではVLSIレイアウトにおける組み立て式配置手法として、ブロック形状に応じた評価値を定義することにより、任意形状ブロックの配置を行う一手法を提案した。本手法の特徴は以下の通りである。

1. 従来手法では難しかった凹部を含む、任意形状ブロックの配置が可能である。
2. 非矩形の領域制限を含む、配置領域を制限しての配

次に50個のブロックを配置する実験例を示す。このとき個対数 M を100、世代数 T を50とした。この例では総セル数は240、そこで15行16列の配置領域制限を設ける。その結果、図11に示したように、50個のブロックすべてを隙間なく配置することができた。なお、この例では凹部を持つブロックが10個存在するが、そのすべてが無駄領域を生ずることなく配置された。また、非矩形の配置領域への配置

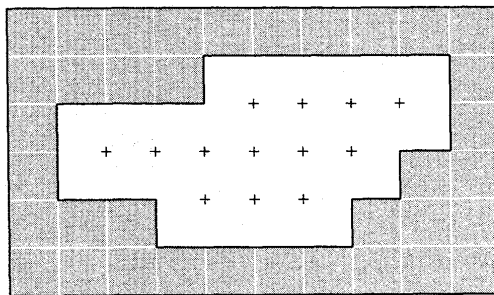


図9. 非矩形の領域制限

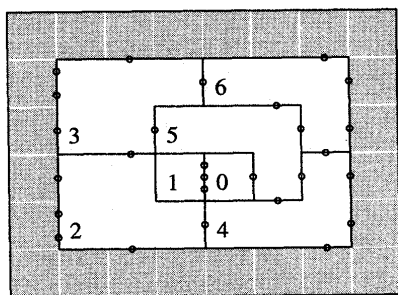


図8. 領域制限があるときの配置 (例1)

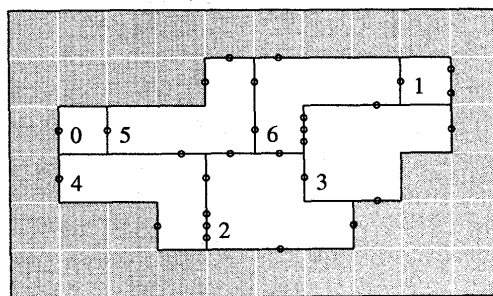


図10. 非矩形の領域制限があるときの配置 (例1)

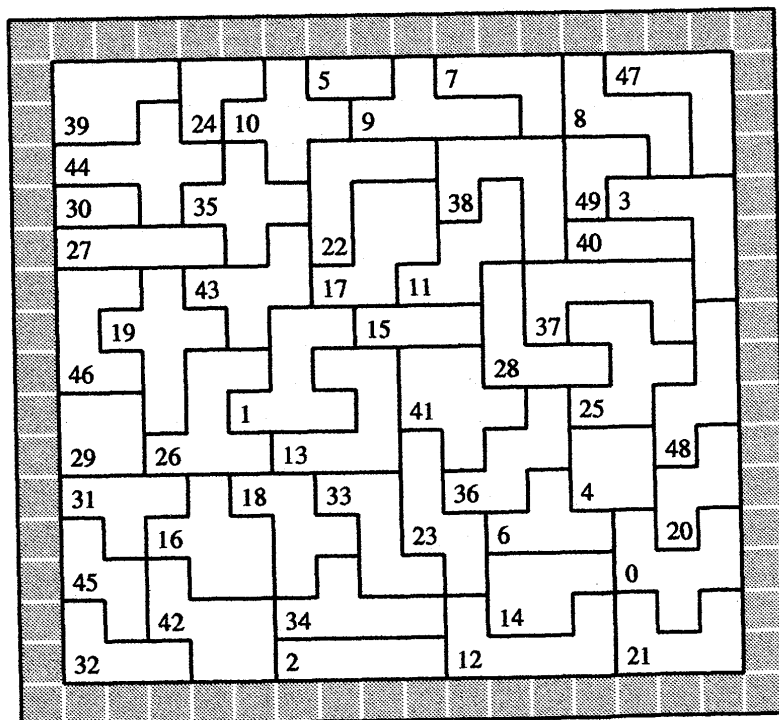


図11. 領域制限があるときの配置 (例2)

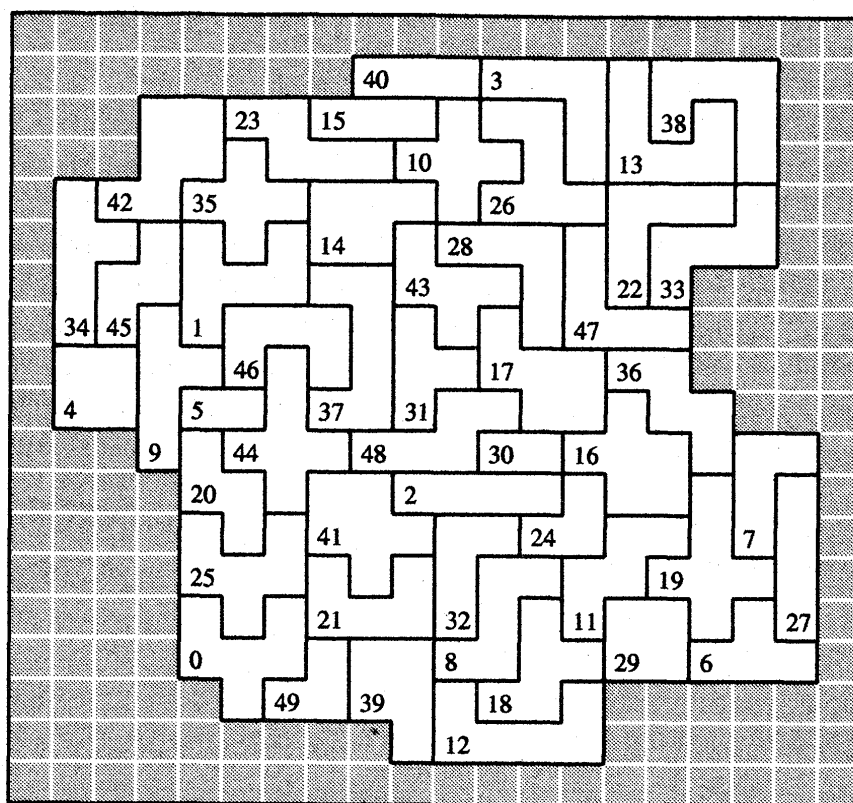


図12. 非矩形的領域制限があるときの配置 (例2)

置が可能である。

3. 配置順序に遺伝的アルゴリズムによる制御を組み合わせることで、組み立て式手法に共通する欠点である配置結果の順序依存問題に対応できる。
4. 従来のコンパクト化手法のための初期配置を与えることに利用可能である。

また、本手法の有用性を確認するため計算機実験を行い、凹部を含む完全な任意形状ブロックの配置や、配置領域に制限を設けての配置も可能であることを示した。

(1998.4.20-受理 1998.7.13-再受理)

文 献

- 1) 渡辺孝博：“LSIレイアウト自動設計の現状と可能性”，信学誌，Vol.76，No.7，pp.774-782，Jul. 1993.
- 2) 佐藤真司，堤 定雄，後藤源助：“マスタスライスセルの自動発生”，信学技報，ICD88-70，1988.
- 3) D.F.Wong and C.L.Liu：“Floorplan Design for Rectangular and L-shaped Modules”，Proc. Intl. Conf. on Computer-Aided-Design，pp.520-523，1987.
- 4) 大村道郎，宮尾淳一，菊野 亨，吉田典可：“VLSIのブロック配置における重なり除去問題”，信学論(A)，Vol.J72-A，No.7，pp.1093-1100，Jul. 1989.
- 5) J.A.Hudson， J.A.Wisniewski and R.C.Peters：“Module Positioning Algorithms for Rectilinear Macrocell Assemblies”，21st Design Automation Conference，pp.672-675，1984.
- 6) 中谷直司，金杉昭徳，森末道忠：“任意形状ブロックの一配置手法”，プリント回路学会第8回学術講演大会講演論文集，pp.85-86，Mar. 1994.
- 7) 北野宏明：“遺伝的アルゴリズム”，産業図書，1993.
- 8) D.Lawrence：“Handbook of Genetic Algorithms”，Van Nostrand Reinhold，1991.
- 9) J.D.ピースリー：“ゲームと競技の数学”，サイエンス社，1992.
- 10) N.Nakaya and K.Kondo：“A Novel Coding Technique for Genetic Placement Method”，Proc. of ITC-CSCC'97，pp.661-664，1997.
- 11) J.Grefenstette， R.Gopal， B.Rosmaita and D.VanGucht：“Genetic Algorithms for the Traveling Salesman Problem”，Proc. of 1st Int. Conf. on Genetic Algorithms and Their Applications，pp.160-168，1985.