設 計 自 動 化 71－2
（1994．6．24）

配線プログラムに適した新データ構造の評価

小島 直仁

株式会社 東芝　研究開発センター
e-mail: kojima@srd.ull.rdc.toshiba.co.jp

VLSI レイアウト CAD の配線プログラムではチップ内のセルや配線を表す図形データを効率よく扱うため Priority Search Tree(以下 PST) などのデータ構造を用いることが必須である。しかし、PST は矩形データを 4 本の構成線分に分解して扱うため計算機上で非常に大きな主記憶を必要とする。そこで本稿では PST を直接矩形が扱えるように拡張した新データ構造を提案する。データ数が n の時、この新データ構造は $O(n)$ のメモリを使用し、平面上で指定点から指定方向にもっとも距離の小さい図形の検索を $O(\sqrt{n}\log n)$ の処理時間で実現する。配線プログラムを実行する上で新データ構造と PST を比較した計算機実験結果とともに、新データ構造の有効性を示す。

A New Data Structure for Routing Programs and its Evaluation

Naohito Kojima

Research and Development Center, Toshiba Corporation
1, Komukai Toshiba-cho, Saiwai-ku, Kawasaki 210, Japan
e-mail: kojima@srd.ull.rdc.toshiba.co.jp

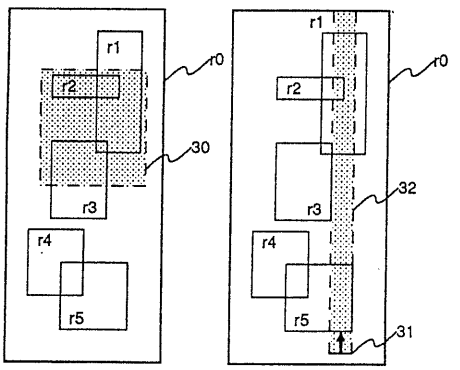
This paper presents a new data structure for handling rectangles improving on the Priority Search Trees(PST). The proposed data structure provides fast geometrical data search and is suitable for various LSI routing methods. The new data structure needs $O(n)$ memory and finds the nearest rectangle to the specified point in a 2-dimensional area in $O(\sqrt{n}\log n)$ time, where n is the number of data in the structure. Experimental results show that the proposed data structure occupies one sixth of the memory area compared to the PST and finds the nearest rectangle to the specified point as fast as the PST.

1 はじめに

VLSIが大規模化するにつれ、VLSIのCADシステムの扱うデータ量も増加の一途を辿っている。VLSIのCADシステムのうち、レイアウト部分が扱うデータの多くを占めるのが矩形を代表とする図形情報である。以後単純に「図形」と記述した場合は矩形を指すものとする。

レイアウト段階では、チップ内のセルの外枠の情報や、配線セグメント、ビアなどのデータが矩形の形でCADシステム内部に保持される。これらのデータはチップを2次元平面(以降xy平面で考える)と仮定した中に分散している図形であり、これらに対する検索はほとんどがxy平面上での図形的な探索である(図1参照)。以後、図形を対象とするデータ検索を「図形探索」と呼ぶ。また、便宜的にxy平面のxの正方向を右、負方向を左、yの正方向を上、負方向を下と定義する。矩形の左下点の座標を(x_{left}, y_{down})、右上点の座標を(x_{right}, y_{up})とする。

レイアウトCADで頻繁に実行される図形探索はおおまかに2種類に分類できる。平面上の図形集合の内、指定された領域と交差する図形を列挙する探索(図1(a)参照)(以後「交差図形探索」)と、指定領域から指定方向に向かって最も距離の小さい図形を求める探索(図1(b)参照)(以後「近接図形探索」)である。



(a) 交差図形探索　　(b) 近接図形探索

図1: 2種類の図形探索

図1のr0で示された矩形領域中にr1～r5の矩形集合が存在する。図1(a)は30で示された矩形領域と交差する図形を求める交差図形探索の例であり、解はr1～r3である。図1(b)は31で示された線分から、図の矢印の方向に最も距離が小さい図形を求める近接図形探索の例であり、解はr5となる。この時図の32で表される矩形領域が、探索の対象となる領域である(以後単に「探索領域」と呼ぶ)。

図形探索の高速化のため、従来より図形データを2分木や4分木構造(Quad-Tree[1, 2]等)などの形に構造化して扱う手法が研究されている。高速な近接図形探索を実現できるデータ構造にPriority Search Tree[3](以後「PST」)があるが、PSTは基本的に平面上の点を扱うデータ構造であり、矩形を扱う場合はそれを4本の線分に分解し、さらに各線分を2本の半直線の形に分解して取り扱う。従ってPSTはひとつの矩形データに対して8個のポインタを作るため、矩形を直接扱える場合と比較して使用メモリ量が非常に大きい。

そこで本論文では、PSTを矩形を直接扱えるように拡張することで使用メモリを約1/6に縮小した図形処理用データ構造(以後「PST-R(PST fot Rectangle)」)を提案し、その有効性を報告する。

2 Priority Search Treeの概要

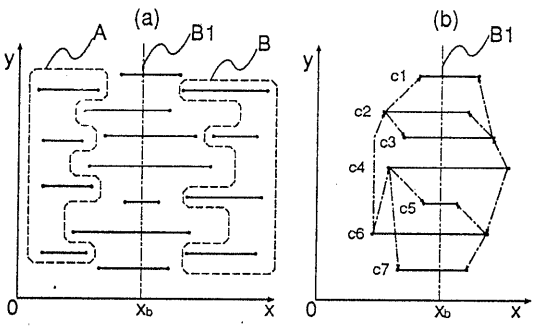


図2: PSTによる線分の構造化

この章ではPSTを用いて矩形集合を構造化して保持し、図形探索をする手法について説明する。PSTでは矩形を直接扱えないので、矩形を水平垂直の各2本の線分に分解して独立に保持する。各線分をさらに2本の半直線として考え、その頂点を管理する。PSTを用いて線分の構造化をする場合、外部木構造と内部木構造の2段階の2分木構造を使う。PSTを用いる線分の構造化とそれに対する探索については[4]に詳しいので、ここでは要点のみを簡潔に述べる。

ここではxy平面内の水平線分集合の構造化について考える(垂直線分の構造化についても同様に行う)。まず外部木構造の作成であるが(図2(a)参照)、垂直の

分割線分B1(x座標をx_bとする)を発生し、それと交差する線分集合Cと分割線分の両側の線分集合A、Bの計3集合に分割する(A、Bに含まれる線分の数の差がたかだか1になるようにする)。この時Cを親とし、A、Bを子孫とする2分木構造を作る。以下、A、Bについて再帰的に以上の処理を繰り返すことで外部木構造が作成される。

外部木構造の各頂点に含まれる線分の集合が内部木構造化の対象となる。各線分を、分割線分の位置で2本の半直線に分解すると、頂点のx座標がx_b未満の半直線の集合とx_b以上の半直線集合であると考えられる。ここでx座標がx_b未満の集合の構造化について説明する(図2(b)参照)。頂点のx座標が最小である半直線をひとつ取り出し(図2(b)ではc6)、それを2分木構造の親とする。残りの頂点をy座標の値でソートし、y座標値がある値(識別値と呼び、2分木構造の各レベルにおける親が保持する)未満と以上である2集合に分割し(図2(b)では{c1 ～ c3}と{c4 ～ c5、c7})それらを子孫として2分木構造を作成する。以下、それぞれの子孫について以上の処理を再帰的に実行することにより内部木構造が作成される。

矩形数がnの時、PSTを用いることで近接図形探索が$O(log^2n)$、交差図形探索が$O(log^2n+j)$(jは解の数)で実行できる。が、ひとつの矩形に対して8個のポインタが必要になるため、計算機上での実行時に大量のメモリを必要とする。レイアウトCADのように巨大なデータを扱うアプリケーションでは、実行時に必要なメモリを最小限にする必要があるので、高速な図形探索を提供するPSTと言えども、今後これを継続使用していくことは困難である。

3 PST-Rの詳細

3.1 PST-Rの特長

PST-RはPSTを矩形を直接扱えるように拡張したもので、PSTと同様に外部木構造と内部木構造の2段階の2分木構造からなる。PST-Rの主な特長は以下の通りである(計算複雑度については後述する)。

1. 矩形データを線分に分解せず、矩形のまま扱うのでPSTと比較して使用メモリが非常に少ない。
2. 偏りの少ない理想的な2分木構造を作るので、時間的無駄のない探索が可能。
3. 木構造の基本的な部分はPSTと同様であり、従って図形探索時間が小さい。
4. 近接図形探索の最大探索距離を指定する(指定された値よりも離れた図形を探索の解としない)ことで、より高速な近接図形探索が可能。

3.2 データ構造の作成

外部木構造作成 PSTでは矩形を水平及び垂直の線分に分解し、それぞれの集合の分割線分は垂直及び水平のみであたが、PST-Rでは水平及び垂直の分割線分を交互に用いる。図3(a)を例にすれば、まずR1を2分割するような垂直方向の分割線分B1をR1内に発生し、B1と交差する矩形集合C{R2 ～ R7}と残りの2分割された矩形集合A{R8 ～ R12}、B{R13 ～ R17}の計3集合に分割する(この時AとBに含まれる矩形数の差はたかだか1になるようにする)。次にB1によって2分割された矩形集合各々について、前記分割処理を今度は分割線分の方向を水平に変えて行う。以上を再帰的に行うことにより、外部木構造が作成される。図3(c)は図3(a)の矩形集合を外部木構造化した例である。

内部木構造作成 外部木構造の各節点に含まれる矩形集合は内部木構造化の対象となる。PST-Rの内部木構造をおおまかに説明すると、矩形の4つの頂点を順に考慮してPSTと同様に内部木構造化したものである。例えば、垂直な分割線分と交差する矩形集合中で右下点のx座標が最大の矩形をひとつ取り出し、残りの矩形を右下点のy座標でソートして2分割する。次に今2分割された矩形集合それぞれについて、今度は左下点のx座標が最小の矩形をひとつ取り出し、以下同様の処理を行う。以降、右上点、左上点、再び右下点の順番で以上の処理を再帰的に繰り返すことで内部木構造が作成される。

図3(b)を例にすると、x_{right}が最大のR4を取り出し、残りをy_{down}でソートして矩形集合D{R6、R7}及びE{R2、R3、R5}に2分割する。D、E各々についてx_{left}が最小のR7、R2を取り出し、残りを前回同様にソートし、2分割する。図3(d) ～ (f)は図3(b)の矩形集合を内部木構造化した例である。

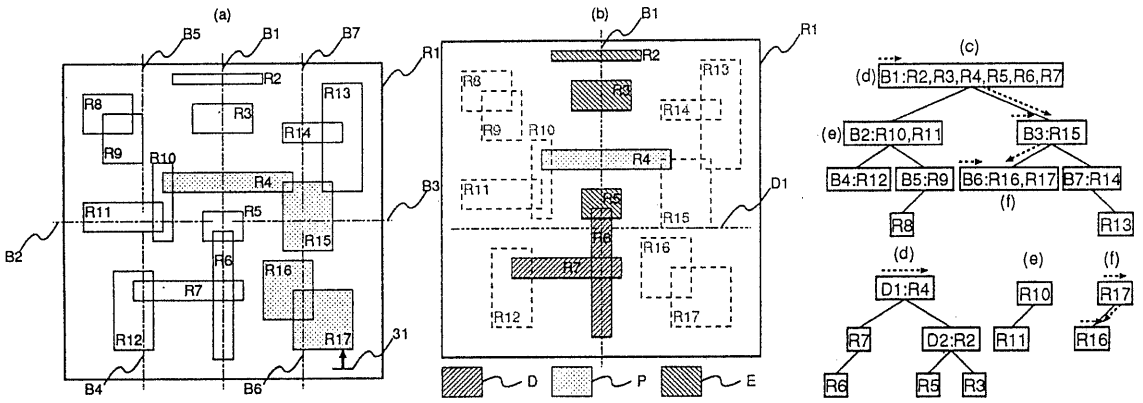


図 3: PST-R による図形の構造化と近接図形探索

3.3 近接図形探索

次に、PST-R を用いて近接図形探索を行う場合の処理について、図 3を用いて説明する。図 3(a) において 31 で示される線分から、図の矢印の方向に向かって最も距離の小さい矩形を求める。

外部木構造内探索 外部木構造内で近接図形探索を行う処理は、大きく分けて (1) 分割線分と平行方向に探索するか、あるいは (2) 垂直方向に探索するかの 2 通りになりこれら 2 種類が順に繰り返される。(1) は PST における外部木構造内探索と同様になる。(2) の場合は、基本的には Quad-Tree における近接図形探索と同様の処理になる。

前記 (1) の例として図 3(a) において線分 31 から上方向へ探索をする場合を例にすると、探索の方向が分割線分 B1 と平行であり、探索領域が B1 と交差しないので、31 から見て B1 の反対側にある矩形集合 A{R8 ～ R12} は探索する必要がない。これ以外の 2 つの矩形集合 (B{R13 ～ R17}、C{R2 ～ R7}) を探索した結果各々得られた解 ('R17' と '解なし') のうち最も 31 との距離が小さい方 (R17) を最終的な解とする。

上記の矩形集合 B に対する近接図形探索探索は、先に述べた (2) の探索の例となり、Quad-Tree と同様に、3 つの矩形集合 A'{R16、R17}、C'{R15}、B'{R13、R14} をこの順に探索の対象とし、A' または C' の中から解がみつかった場合は B' を対象とする探索をしない。この場合解として R17 がみつかっているので B' 内の探索をせずに解を R17 に決定する。

以上 2 種類の探索を交互に繰り返すことで、外部木構造内の近接図形探索が実現できる。

内部木構造内探索 内部木構造内で近接図形探索を行う処理も外部木構造と同様に大きく分けて、外部木構造作成時の分割線分と (1') 平行方向に探索するか、(2') 垂直方向に探索するか (図 4の S0、S1 からの探索) の 2 通りになる。(1') は PST における内部木構造内探索とほぼ同様であり、(2') については、前節での (1) とほぼ同様である。但し PST-R の内部木構造では矩形集合分割時のソートのキーとなる点として矩形の 4 頂点を順に用いているので、探索の解を得るために木構造内の探索が 2 つに枝別れする必要が生じるが、うち一方は最悪でも 2 世代の探索で打ち切られるので内部木構造内図形探索に関しては PST と大きく変わるものではない。

内部木構造内図形探索では、探索の最大奥行きを指定することで効率良く探索することができる。例えば図 4S0 の探索の最大奥行きの x 座標が R4 の x_{right} より大きければ、図の矩形の中に解がないことは最初に R4 を参照するだけで認識できる。

3.4 交差図形探索

次に、PST-R で構造化した矩形集合に対して交差図形探索を行う場合の処理について説明する。

ある線分からの探索の奥行きつきの近接図形探索は、一辺がその線分の長さ、あと一辺が探索の奥行きに等しい長方形領域と交差する全ての図形のうち、線分からの距離が最も小さいものをひとつ選ぶ処理であるが、交差図形探索の場合は指定領域と交差する図形

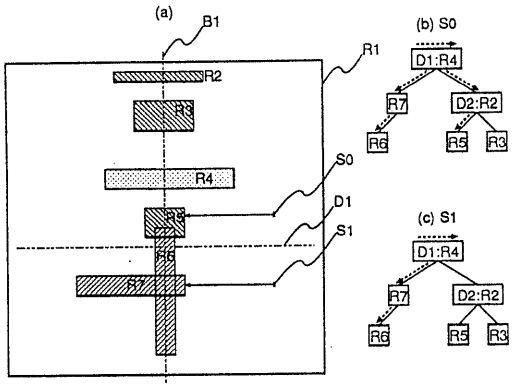


図 4: PST-R による近接図形探索 (内部木構造)

を全て列挙する。従って、外部木構造及び内部木構造内の探索の手法は本質的に近接図形探索と大きく変わるものではない。

3.5 計算複雑度

本節では、扱う図形の数を n とした時の計算複雑度について述べる。まず使用メモリ量について述べ、以下データ構造作成、近接図形探索、交差図形探索の時間複雑度について説明する。

ここで、n 個の矩形が、m 個の内部木構造 (=外部木構造のノード数) に分散して含まれ、それぞれの内部木構造が自然数 $k_i (0 \leq i \leq m-1)$ 個の矩形を含むものとする。この時 $n = \sum_{i=0}^{m-1} k_i$ であり、また明らかに $m \leq n$ 及び $k_i \leq n$ である。

3.5.1 使用メモリ量

使用メモリ量は、内部木構造のノード数が n、外部木構造のノード数が明らかに n 以下であるから、$O(n)$ である。

3.5.2 データ構造の作成

データ構造作成時間は、(1) すべての図形を外部木構造のノードに振り分ける処理時間と、(2) 外部木構造の各ノードにおいて、内部木構造を作成する処理時間からなる。

上記 (1) の時間複雑度は、外部木構造の世代数が $O(\log n)$、それぞれの世代での子孫への図形の振り分け処理時間が $O(n)$ であるから、$O(\log n) \times O(n) \rightarrow O(n \log n)$ である。

上記 (2) の時間複雑度について考える。内部木構造内の矩形の数を k とする ($k \leq n$)。内部木構造の世代数が $O(\log k)$ である。それぞれの世代での処理は分割線分から最も離れた位置を占める矩形を取り出す処理が $O(k)$、矩形集合をソートするのに $O(k \log k)$ であるからそれらを加えて $O(k \log k)$ である。従ってひとつの内部木構造を作成する時間複雑度は $O(\log k) \times O(k \log k) \rightarrow O(k \log^2 k)$ となる。(2) 全体の時間複雑度を求めるには、以上を内部木構造の数だけ繰り返した場合を想定すればよい。前述した通り $n = \sum_{i=0}^{m-1} k_i$ であるから、(2) 全体の時間複雑度は、

$$\sum_{i=0}^{m-1} k_i \log^2 k_i \leq \sum_{i=0}^{m-1} k_i \log^2 n = n \log^2 n$$

より、$O(n \log^2 n)$ となる。

(1)、(2) の時間複雑度から、データ構造作成全体の処理時間の時間複雑度は $O(n \log^2 n)$ となる。

3.5.3 近接図形探索

図形の存在する領域全てを探索の対象とし、探索の奥行きを図形の存在する領域の端までとする。また、探索の始点となる線分の長さが図形の存在する領域の大きさに対して充分小さいものとする[1]。

外部木構造内の探索の時間複雑度 まず外部木構造内の探索の時間複雑度について考える。探索の始点となる線分の長さが充分小さい場合、外部木構造内のルートノード (root node) からの探索は、近接図形探索の方向が外部木構造作成時の分割線分の方向と (1) 平行である場合、及び (2) 垂直である場合の 2 通りとなる。

(1) の場合 (図 3(a) において探索の方向と分割線分 B1、B6 が平行)、木構造内の探索は二方向に別れたりせずに末端方向へ進んでいく。(2) の場合 (図 3(a) で探索の方向と分割線分 B3 が垂直) は、最悪の場合は 2 つの子孫両方の木構造内を探索する。

外部木構造内の近接図形探索では上記 2 種類の探索が交互に繰り返されるので、外部木構造内のノードの数は m であるから、外部木構造内の近接図形探索の時間複雑度は $O(\sqrt{m})$ である。

[1] 探索の始点となる領域の大きさが図形の存在する領域に等しい場合、探索の時間複雑度は $O(n)$ となり、実用上の意味がなくなる。

内部木構造内の探索の時間複雑度　次に内部木構造内の探索について考える。まず内部木構造内のルートノードからの探索は近接図形探索の方向が外部木構造作成時の分割線分の方向と(1')平行である場合(図3(a)参照)、及び(2')垂直である場合(図4参照)の2通りとなる。

前節で述べた通り、(1')はPSTにおける内部木構造内探索($O(logn)$(n:　図形数))とほぼ同様であり、(2')については、外部木構造内探索(1)($O(logn)$(n:　図形数))とほぼ同様の処理を行う。ここでPST-Rの内部木構造では矩形集合を直接扱うため、位置が一部重なっているような複数の矩形を完全に2分割することができない。そのため、探索の解を得るために木構造内の探索が2つに枝別れすることがあるが、2分割のもととなる座標値は2回ずつ交互に入れ替わるので、木構造内の探索が2つに枝別れするのは最悪でも連続2世代のみである。従ってこのことは計算複雑度に影響を与えず、PST-Rの内部木構造内近接図形探索の計算複雑度はPSTや、外部木構造内探索(1)と等しくなる。よって、内部木構造に含まれる図形数をkとする時、近接図形探索の時間複雑度は$O(\log k)$である。

木構造全体での時間複雑度　外部木構造内の探索の時間複雑度は$O(\sqrt{m})$である。外部木構造の各ノードに含まれる内部木構造内の探索の時間複雑度は$O(\log k)$であるから、近接図形探索全体の時間複雑度は$O(\sqrt{m})\times O(\log k) \rightarrow O(\sqrt{m}\log k)$となる。

あきらかに$m \leq n$、$k \leq n$であるから、近接図形探索全体の時間複雑度は$O(\sqrt{n}\log n)$となる。

3.5.4　交差図形探索

与えられた図形集合の中から、指定領域と交差するものを全て列挙する探索である。図形の存在する領域全てを探索の対象とし、近接図形探索と同様に、指定する領域の大きさが図形の存在する領域の大きさに対して充分小さいものとする。

まず、交差図形探索の解が1個見つかる場合の時間複雑度について考え、次に解がj個見つかる場合の時間複雑度を求める。

外部木構造内の探索の時間複雑度　PSTと同様の探索をするので、時間複雑度も同様に考える。従って外部木構造内のノードの数をmをすると、外部木構造内の交差図形探索の時間複雑度は$O(\log m)$である。

内部木構造内の探索の時間複雑度　前節で述べた近接図形探索と同様に考えることができるので、内部木構造に含まれる図形数をkとする時、交差図形探索時間複雑度は$O(\log k)$である。

木構造全体での時間複雑度　外部木構造内の探索の時間複雑度は$O(\log m)$であり、内部木構造内の探索の時間複雑度は$O(\log k)$であるから、交差図形探索全体の時間複雑度は$O(\log m) \times O(\log k) \rightarrow O(\log m \log k)$となる。

あきらかに$m \leq n$、$k \leq n$であるから、交差図形探索全体の時間複雑度は$O(\log^2 n)$となり、以上は解が1個だけ見つかる場合の時間複雑度であるから、解がj個存在する場合の時間複雑度は$O(\log^2 n + j)$となる。

4　計算機実験結果

4.1　はじめに

PST-Rと、PSTの2種類のデータ構造を用いて、同じチップデータに対する配線処理を行い、使用メモリの量やデータ構造作成、図形探索に費やした処理時間を測定し、比較する。

表1: PST-RとPSTの計算複雑度の比較(扱う図形数: n、交差図形探索の解の数: j)

	PST	PST-R
使用メモリ量	$O(n)$	$O(n)$
データ構造作成	$O(n\log^2 n)$	$O(n\log^2 n)$
近接図形探索	$O(\log^2 n)$	$O(\sqrt{n}\log n)$
交差図形探索	$O(\log^2 n + j)$	$O(\log^2 n + j)$

PST-RとPSTの計算複雑度を比較すると、表1のようになる。PSTは矩形を8つの半直線に分解して扱うため、1つの矩形を指すポインタが8個になるのに対して、PST-Rは1つの矩形を指すポインタが1個でおさえられる。扱う図形数をnとすると使用メモリ量はどちらもnに比例するが、実際の使用量はPST-Rの方が非常に小さい。

表2: LERに入力された図形数と使用メモリ量(Kbytes)、データ構造作成時間(min)、図形探索処理時間(min/1,000回)及びLER全体の処理時間(min)

データ名	図形数	使用メモリ量		データ構造作成		近接図形探索		交差図形探索		LER全体	
		PST	PST-R	PST	PST-R	PST	PST-R	PST	PST-R	PST	PST-R
A	15534	2106	374	2.57	1.58	0.12	0.22	0.070	0.106	32.74	33.48
B	34423	4633	783	6.20	4.43	0.13	0.27	0.043	0.095	76.29	86.86
C	40970	5456	820	7.48	4.95	0.10	0.25	0.043	0.140	129.90	173.19
D	68801	9081	1505	13.55	9.07	0.12	0.26	0.070	0.097	186.77	215.74
E	186969	24218	3735	38.28	28.54	0.17	0.33	0.055	0.110	135.83	140.96

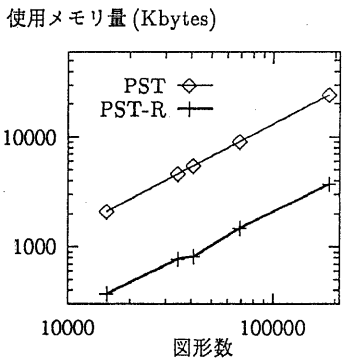


図5: LERに入力された図形数と使用メモリ量

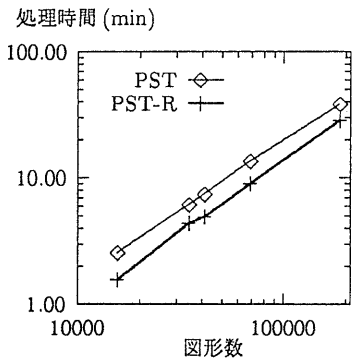


図6: LERに入力された図形数とデータ構造作成時間

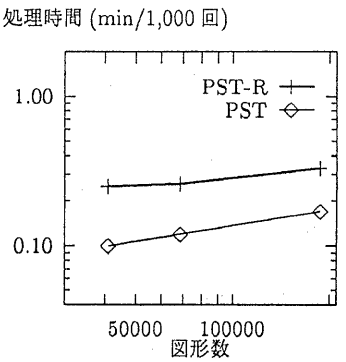


図7: LERに入力された図形数と近接図形探索処理時間

時間複雑度については、近接図形探索においてPST-Rの時間複雑度が大きくなっているが、これは起こり得る最悪の場合を想定しているものであって、VLSIレイアウトで扱う図形データのように、指定領域内に領域の大きさと比較して充分小さい図形が平均して多数分散しているような場合では時間複雑度は概ねPSTと等しくなるものと思われる。

実験に使用した計算機はSun SPARCstation10(CPU数:1)(SPECint92: 65.2)、使用Cコンパイラは GNU Cである。各データ構造の使用メモリは、その全てがCの構造体の確保であるため、確保した構造体の大きさと個数を乗じて計算した。処理時間の測定にはgprofを使用した。

使用アプリケーションはゲートアレイ(Sea of Gates)用クロック配線プログラム[5]である。このクロック配線プログラムの配線エンジンLER(Line Expansion Router) [6, 7, 8]の配線処理にPST-RとPSTをコンパイル時に切替えて使用できるようにし、使用メモリ量と処理時間を測定した。

使用したデータは5種類のSea of Gatesデータであり(便宜的にA～Eとする)、内2データが2層(A、B)、3データが3層(C、D、E)である。全セル数はAからEへ順に多くなっており、LERに入力される図形数はこの値に比例する。

4.2 使用メモリ量

LERに入力された図形数と使用メモリ量を表2と図5に示す。図5より、両データ構造とも使用メモリ量は概ね図形数に比例していることがわかるが、PST-Rのメモリ使用量はPSTの1/6程度であり、VLSIレイアウトシステムに組み込む上では非常に有効である。

4.3 データ構造作成と図形探索の処理時間

表2及び図6を見てわかるとおり、データ構造作成時間は両データ構造とも概ね図形数に比例しているが、

PST-R の方が PST よりも処理時間が小さい。これは、PST が矩形データを 8 つの半直線で表すことで取り扱うデータ量が増加しているのに比べて、PST-R は矩形をそのまま扱うのでデータ量が少ないことによる。

近接図形探索については、3 層データについてのみグラフ化した。データによって、図形探索の回数が異なるので、探索 1,000 回あたりの処理時間を求めて比較することにした。図 7 を見てわかるとおり、PST と PST-R でグラフの傾きはあまり変わらず、むしろ PST-R の方が傾きが小さい。時間複雑度では PST-R は PST よりも不利な結果が得られたが、実用上(特に大規模なデータを扱う VLSI レイアウト)では大きな問題はないものとみなすことができる。

図形探索のグラフの傾きが他のグラフと比較して急でないのは、大規模なデータで配線を行う場合でも個々の配線は極小さい領域内で行われ、図形探索の範囲も小さい領域にしぼられるからであるものと考えられる。

4.4 アプリケーション全体の処理時間

5 種類のデータを使った場合における、データ構造の違いと LER の処理時間の関係を表 2に示す。PST-R を使っているものは PST のものよりも若干処理時間が大きいが、しかしその差は数％～20% 程度である。これは PST-R のデータ構造作成時間が小さいことと、矩形を直接扱えることで探索回数自体を少なくできることによる。

5 おわりに

VLSI レイアウトシステムにおける配線プログラムに最適なデータ構造について報告した。配線プログラムに PST-R を使用することで、データ検索が高速な PST と比較して処理時間が若干増加するものの、必要な主記憶を大幅に小さくする (PST 使用時の約 1/6) ことができる。

クロック配線システムの動作時間のうち配線処理時間は多くを占めているが、PST-R を使用することでそれを十分小さい範囲に抑えることができ、その結果、我々が開発中のクロック配線システムは現在一般的に使用されている同様の製品よりも処理時間が小さくなっている。

先の計算機実験では LER に PST と PST-R の 2 種類のデータ構造を組み込んで実験をしたが、LER は配線処理部分とデータ構造部分のモジュール分離がされているので、比較的簡単にデータ構造を切替えることができる。本報告では、現在考えられる最適のデータ構造について報告したが、今後、PST-R の改良点を発見できた場合はもちろん、将来さらによいデータ構造を考案した場合はすぐに LER に組み込んでシステム全体の性能アップを図りたい。

参考文献

[1] Samet H.: "The Design and Analysis of Spatial Data Structures", Addison-Wesley Publishing Company, Inc.(Apr. 1990).

[2] Samet H. : "Applications of Spatial Data Structures", Addison-Wesley Publishing Company, Inc.(Jun. 1990).

[3] McCreight E.M. : "Priority Search Trees", SIAM J.Comput., **14**, No.2, pp.257–276(May. 1985).

[4] Ohtuski T. et al.n: "Layout Design and Verification", North Holland, pp.313–316(1986).

[5] 高野, 南, 小島, 三橋 : "ディレイ・スキュー最小化のための線幅最適化クロック配線手法", 信学技法, **VLD93-9**, (Jun. 1993).

[6] Heyns W., Sansen W. and Beke H. : "A Line-Expansion Algorithm for the General Routing Problem with a Guaranteed Solution", Proc. 17th DA Conf., pp.243–249 (Jun. 1980).

[7] 小島, 佐藤, 大附: "線分展開法の改良とその評価", 情処学研報, **DA89**, No.6, pp.1–8(Jul. 1989).

[8] 小島, 山田, 佐藤, 大附: "線分展開法の 2 層配線への拡張とその評価", 情処学研報, **DA91**, No.3, pp.1–8(Jul. 1991).