設 計 自 動 化 61－1
（1992. 2. 21）

レイアウトデータ管理システム上の各種データ構造の評価

池田 泰人　池田 栄一郎　粟島 亨　久保田 和人　佐藤 政生　大附 辰夫

早稲田大学理工学部

あらまし

電子回路のレイアウト設計は，配置，配線，設計規則検証といったいくつかの工程から成り立っているが，これらの工程で必要となる基本的な図形処理を，共通化した関数として用意し，プログラム作成効率及び処理効率を高めるという目的でレイアウト・データ管理システム(DMS)が提案されている．本稿では，DMS上の基本的な図形処理関数をいくつかのデータ構造を用いて実装し，線分展開法に基づいた多層配線プログラムを実行することによって，実際的な見地から，各データ構造の性能を定量的に評価する．

Experimental Estimations among the Data Structures Implemented on Layout Data Management System

Yasuto Ikeda, Eiichiro Ikeda, Toru Awashima, Kazuto Kubota,
Masao Sato, Tatsuo Ohtsuki
School of Science and Engineering, Waseda University
3-4-1 Ohkubo, Shinjuku-ku, Tokyo 169,Japan

Abstract

Layout design process consists of some processes such as floorplaning, routing, design rule verification, etc. Their processes need same geometrical search functions. Layout Data Management System has been proposed to facilitate developing layout design tools and improve their performance. Several geometrical search functions are implemented on DMS using some data structures. In this paper, practical performance of each data structure is estimated by executing a gridless router implemented on Layout Data Management System.

1．はじめに

電子回路のレイアウト設計は，配置，配線，コンパクションといった一連のアプリケーション・プログラム（以下単にアプリケーションと呼ぶ）を用いて行われる．これらのアプリケーションはレイアウト・データの参照・更新を基本として成り立っている．また，設計の過程においては各アプリケーション間で，レイアウト・データの授受が頻繁に行われるのが普通である．したがって，レイアウト設計の作業効率を上げるためには，いかにしてレイアウト・データを管理するかがひとつの鍵となる．このような背景のもと，我々はレイアウト・データ管理システム（Data Management System 以下ＤＭＳと呼ぶ）を提案してきた．[1] ＤＭＳはレイアウト・データを対象とした一種の共有データベースシステムであり，以下に挙げるような特徴を有している．

（1）レイアウト・データ形式の統一

レイアウト・データはＤＭＳ内に一括して，一つの形式で保持される．これにより，アプリケーション間のデータ変換の手間がなくなる．

（2）レイアウト・データの保護

アプリケーションはシステムの許可がない限りレイアウト・データを更新することを許されず，参照することのみが許されている．これは，不用意にデータを更新し，破壊してしまうことを防ぐためである．

（3）レイアウト・データに対する操作仕様の標準化

レイアウト・データに対する操作のうち基本的なものはアプリケーション間で共通であることが多いため，あらかじめ標準化された関数群として提供する（ＤＭＳ関数と呼ぶ）．これにより，アプリケーション開発の負担が軽減される．

レイアウト設計は図形的な最適化問題と考えられ，用いられるアプリケーションはその大部分が図形的な処理から成り立っている．このような観点からＤＭＳではレイアウト・データに対する図形的操作の効率化に特に配慮がなされている．レイアウト・データは２次元的な図形データとして表現されるが，このような図形データを管理するデータ構造は多数提案されており[2-6]，これらのデータ構造のいずれを用いてもＤＭＳ関数を実現することが可能である．しかしながら，図形的操作の性質によってふさわしいデータ構造は異なると考えられるので，一連のＤＭＳ関数を各種のデータ構造を用いて実装し，その効率を比較，評価することは有意義であろう．データ構造の評価はＤＭＳという統一的な環境下で行われるため，種々の条件を合わせることが容易であり，結果として比較的公平な評価が期待できる．

我々は既に更新を伴わない静的なデータに対する各種データ構造の評価を行っている．[7]しかし，現実のアプリケーションの場合，実行中に絶えず動的にデータが更新されるのが普通であろう．そこで本稿では，ＤＭＳを利用して実装された多層の配線アプリケーション[8]を実行することによって，各種データ構造をより実際的に評価する．評価したデータ構造は，バケット，４分木[2,3,4]，フィールド・ブロック[5]，ヒープ探索木[6]の４種類である．

2．レイアウト・データ管理システム（ＤＭＳ）

ＤＭＳは，レイアウト・データとそのレイアウト・データを参照・更新するためのＤＭＳ関数群から構成されており，レイアウト・データは部品やセルの形状および機能を記述したライブラリデータ，ネットリスト，さらに配線禁止領域や配線線分，端子，ビアといった図形データから成り立っている．ＤＭＳ関数としては，「ある端子に接続されるべきネットを特定する」といった，論理的な情報を得るための関数と，「配線領域内のある図形から最も距離の近い図形を求める」といった，図形的な情報を得るための関数が用意されている．ＤＭＳ関数の仕様はデータの管理構造に依存しない標準化されたものであるので，各種アプリケーションはデータの管理構造に対して注意を払う必要はない．

図１にＤＭＳを核としたＣＡＤシステムの構成例を示す．通常ＤＭＳ関数はＤＭＳ内部で管理されているレイアウト・データを処理の対象とするが，アプリケーション自身が生成したデータを処理の対象とすることもできる．これは，処理の過程で生じた補助的なデータも含んでいる．アプリケーションはＤＭＳ内部のレイアウト・データをＤＭＳ関数を用いて自由に参照することが可能である．一方，ＤＭＳ内部のレイアウト・データに対する更新操作はデータの破壊を防ぐ意味からも，ユーザーの確認のもとでのみ行われるべきであろう．このような機構を実現し，システム全体の動作を制御するのがシステムコントローラ部である．

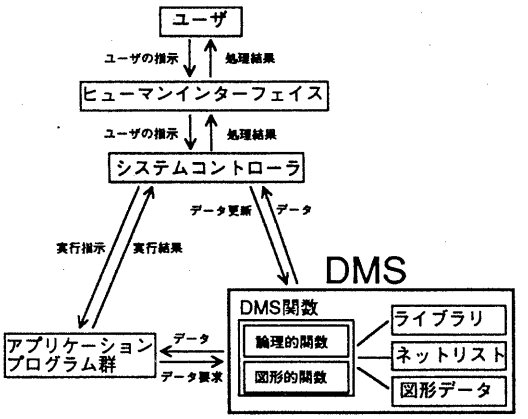

図1　DMSを核としたCADシステム

本稿では，データ構造の違いによる図形探索の効率を比較・検討することを主な目的としているため，次節以降では図形データとそれを探索するDMS関数（以下単にDMS関数といった場合はこのような関数を指すものとする）を実現するためのデータ構造についてのみ議論する．

3．図形データ管理構造

2次元の図形データを管理するデータ構造は，管理の基本単位が線分であるものや，多角形であるものなどさまざまである．そこで，図形データの表現方法として一種の階層表現を用いることにした（図2参照）．

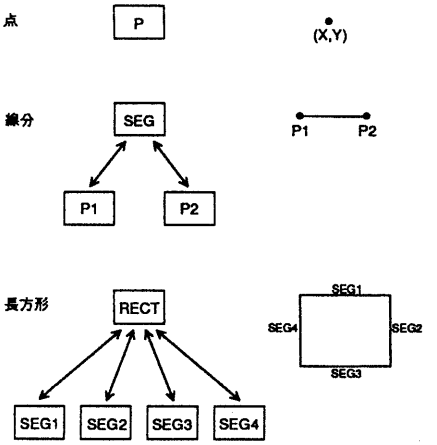

図2　階層的な図形表現

これは，最も基本的な図形単位を点とし，線分は2個の点によって，さらに多角形は線分の集合として表現するという方法である．この表現方法を採用することによって，一つの図形は点としても線分としても多角形としても認識できるようになり，データ構造による管理単位の違いを吸収することができる．また，データ管理構造は，図形データそのものではなく，図形データの実体を指す参照ポインタを管理することとした．これは，図形データの実体とその管理構造を明確に分離することによって，データ管理構造の置換が容易になるからである（図3参照）．

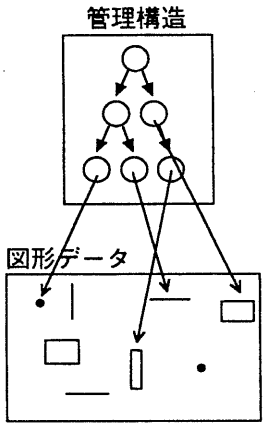

図3　図形データと管理構造

以下では，4節で比較実験を行うデータ構造について簡単に紹介する．

3．1　バケット

バケットは，全体の領域を図形データとは独立にN×Nの格子に分割し，それぞれの格子（これをバケットと呼ぶ）に，その格子と交差する図形を対応させて管理するデータ構造である．これらの図形は各バケットをヘッダとする連結リストで保持される．複数のバケットに交差する図形はそれぞれのバケットに重複して保持される．バケットそのものは配列で実現されており，バケットの図形探索の時間複雑度は

$$O(k \cdot n / N \times N)$$

となる．ここで，kは1個あたりのバケットに含まれるデータの個数である．いまバケット数をnに比例する程度とすれば，ほぼ定数時間での図形探索が実現で

きる．また，空間複雑度は

$O(n+N\times N)$

となる．

3．2　4分木[2,3,4]

4分木(Quad tree)は，Finkel,Bentleyらによって提案された2次元データを管理するデータ構造である．4分木によって多角形データを管理するには，領域を各副領域内に高々1つの多角形しか含まれなくなるまで再帰的に4分割する．多角形は自身を含む副領域（クワッドと呼ぶ）に対応する4分木の葉節点によって管理される．ただし，扱う図形データが膨大な場合，クワッド中に存在する多角形の数があるしきい値S以下になった時点で分割を停止するのが一般的である．4分木は，分割線と交差する多角形の保持の方法でいくつかに分類できるが，ここでは，多角形をその外包矩形の左下の点が含まれるクワッドで管理する改良型4分木を用いている．改良型4分木の図形探索の時間複雑度は

$O(\log_4 n/S+S)$

空間複雑度は

$O(n/S+n)$

となる．

3．3　フィールドブロック[5]

フィールドブロックは，鈴木らによって対話型ＣＡＤシステムのデータ管理のために提案されたデータ構造である．基本的には，領域上の図形データを一定の規則のもとにM個にグループ化し，一つのグループに属するデータを，そのデータを囲む外包矩形（フィールドブロックと呼ぶ）によって管理する．フィールドブロックの図形探索の時間複雑度は

$O(M+\ln/M)$

となる．ここで1は探索の候補となったフィールドブロック数であり通常定数個であると仮定してよい．ここでMを$\sqrt{n}$程度に選ぶことができれば，時間的複雑度をO（$\sqrt{n}$）に近づけることができる．また，空間複雑度は

$O(M+n)$

となる．

3．4　ヒープ探索木[6]

ヒープ探索木は，McCreightによって提案された2次元平面上の点集合を管理するためのデータ構造であるが，次のようにして水平，垂直の線分集合を管理することができる．たとえば，水平線分集合を管理する場合は以下のようにする．まず領域を垂直線分によって再帰的に分割していき（この垂直線分は2分木で管理する．first tree），その垂直線分と交差する水平線分を左端点と右端点に分け，それぞれの端点を別々のヒープ探索木で管理することとし(second tree)，垂直線分を表すノードからこの二つのヒープ探索木にポインタを持たせる．ヒープ探索木の線分探索の時間複雑度は

$O(\log_2 n)$

空間複雑度は

$O(n)$

となる．

4．アプリケーションによるデータ構造の評価

データ構造の評価にあたってはＤＭＳを利用して実装されたアプリケーションとして改良線分展開法に基づいた多層配線プログラムを用いた．この多層配線プログラムを前述の4種類のデータ構造によってそれぞれ実行し，処理時間及び使用メモリについての比較評価を行った．以下ではまず多層配線プログラムの処理内容を簡単に説明し，引き続き評価実験について述べることにする．

4．1多層配線プログラム

評価実験に用いた多層配線プログラムは，改良線分展開法と呼ばれる逐次配線手法である．その処理は基本的に図形的な操作から成り立っており，格子構造に依存しないグリッドレスルータである．以下にアルゴリズムのおおまかな流れを示す．

領域中に始点Ｓと終点Ｔが与えられているとする．本アルゴリズムは，ＳからＴに向かって配線領域を図形的に探索することで径路を得るものである．具体的には，まずＳが配線領域中にある場合にはＳの上下左右の4方向に，Ｓが禁止領域の境界上にある場合は禁止領域と反対の方向に，AL(active line)と呼ばれる線分を発生する．次にこのALをその線分の向きと直角方向に障害物にあたるまで展開し，領域探索を行なう．このとき探索した領域はAR(active rectangle)と呼ばれる矩形として保持する．このARは，異なる層へ向かっての領域探索に用いられる．さらにARの外周にそって必要に応じて新たなALを作成し，同層内での領域探

索を行う．このようなALとARの展開をあらかじめ定義されたコストにしたがってTを発見するまで繰り返す．Tが発見された時点で，バックトレースを行ない径路を得る．以上の説明からも分かるように，本アルゴリズムは以下に挙げる２つの基本的な図形的探索によって構成されている．

（1）線分隣接探索（図4）

質問線分qが与えられたときに，qから指定方向にあり，かつ，qに一番近い線分を報告する．

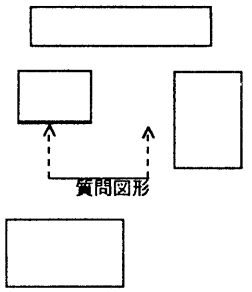

図4　線分隣接探索

（2）領域探索（図5）

質問領域rが与えられたときに，rと交差する図形をすべて報告する．

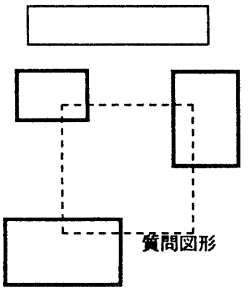

図5　領域探索

この２つの探索操作は最も基本的なＤＭＳ関数として用意されている．

４．２　実験結果

ＤＭＳおよび多層配線プログラムはSparc Station2 (27.5MIPS)上にＣ言語によって実装されている．実験は，12,000×12,000の配線領域内に乱数によって発生させた５組の２端子対を逐次配線するという処理を基本単位として行った．配線層は２層であり，配線方向は縦横原則にしたがっている．配線領域内にはやはり乱数によって生成した矩形の障害物が存在する．実験用のテストデータは禁止領域数が8，32，128であるようなものを５例ずつ作成し，それぞれのデータに対して基本処理を実行した．全ての結果は，禁止領域の数が同じデータ毎に集計して平均値によって評価した．以下では便宜上，禁止領域数8，32，128のデータに関する集計結果をそれぞれデータsmall，medium，largeに関する結果として示すことにする．配線プログラムの実行中は図形データの挿入・削除が頻繁に行われるが，そのうち約60%は探索のために生成されたALやARの挿入・削除であった．表１に基本処理実行中にＤＭＳによって保持された図形数の最大値を示す．

表1　ＤＭＳによって保持された図形数の最大値

データ	最大図形数
small	528
medium	872
large	2118

前述したように，線分展開法は線分隣接探索と領域探索によって実現されている．図６にプログラム実行時の各探索の頻度を示す．図からわかるように，線分隣接探索が全探索の80%以上を占めている．

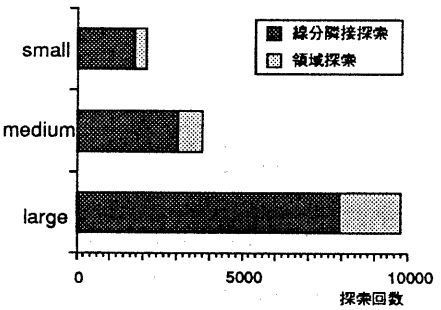

図6　配線処理時の各探索の頻度

４．３　各データ構造の評価

比較したデータ構造の中には固有のパラメータによってその挙動が大きく変化するものがあるが，実験ではデータによらず以下のように一律に設定した．

・バケット
配線領域を10×10のバケットに分割した．
・４分木
しきい値Ｓを10とした．
・フィールドブロック
フィールドブロック内の図形の最大数を25とした．

図７～９に各データに対して配線処理を実行した時の，処理時間を比較した結果を示す．いずれの場合もデータの構造への挿入，構造からの削除，線分隣接探索，領域探索といった図形的な処理が全体の80%程度を占めていることがわかる．以下，各処理ごとに評価する．

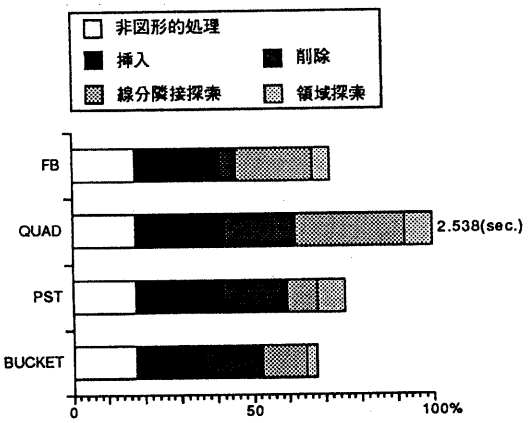

図７　各処理時間の評価(small:配線禁止領域数８)

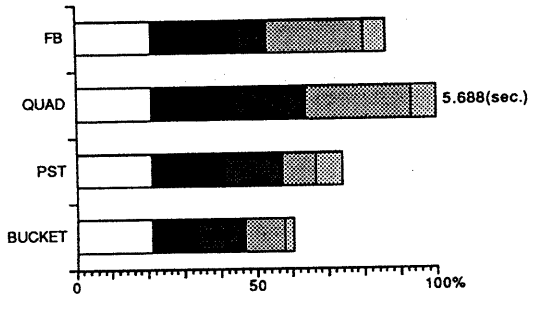

図８　各処理時間の評価(medium:配線禁止領域数３２)

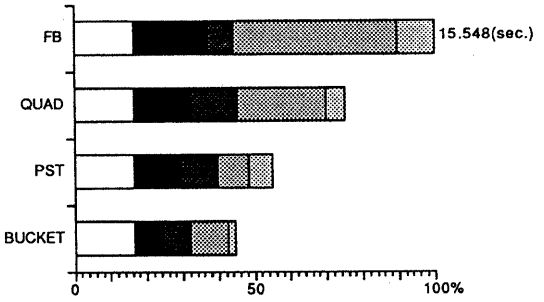

図９　各処理時間の評価(large:配線禁止領域数１２８)

・挿入
いずれの場合も，バケットが最も短い時間で処理を行っている．フィールドブロック，４分木，ヒープ探索木はほぼ同等の性能であるといえる．

・削除
４分木，ヒープ探索木，バケットとも挿入と同程度の手間を要している．一方フィールドブロックは挿入の場合の数分の１程度で処理を行っている．前述したように，配線プログラムは径路を探索する過程で多くのAL,ARを生成している．これらは，径路が確定した後一括して削除されるが，このとき単純な構造を持つフィールドブロックが最も効率よく処理を行うことができるのであろう．

・線分隣接探索
全般に，ヒープ探索木とバケットがほぼ同等でフィールドブロックや４分木の３分の１程度の時間で処理を行っている．ただしフィールドブロックはlargeに関して処理効率が急激に低下している．これはパラメータの設定がこの規模のデータに対して適切でなかったためと考えられる．

・領域探索
いずれの場合でもバケットが最も短い時間で処理を行っている．フィールドブロック，４分木，ヒープ探索木はほぼ同等で，バケットの４～５倍の時間を要している．

・メモリ使用量
ヒープ探索木は，絶対的なメモリ量，問題の規模に対する増加率ともに他のデータ構造に比べて著しく大きい．ついで，４分木，バケット，フィールドブロッ

クの順となる．特にフィールドブロックは構造が単純なこともあって，使用メモリは非常に少なくなっている．

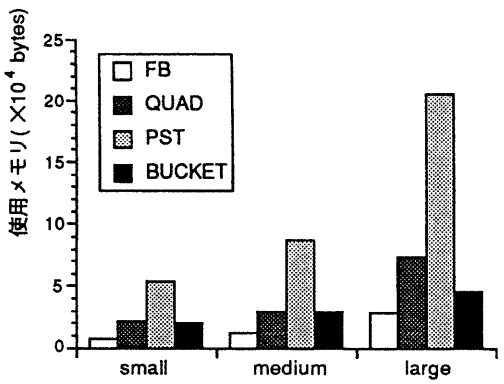

図１０　各データ構造のメモリ使用量

５．おわりに

本稿では，これまで我々が提案してきたレイアウト・データ管理システムが提供する各種の図形探索関数をフィールドブロック，４分木，ヒープ探索木，バケットの４種のデータ構造で実装し，改良線分展開法に基づく多層配線アプリケーションを実行することによって，実際的な見地からデータ構造の定量的な評価を行った．実験の結果をみると，処理全体に占める図形的処理の割合はおおむね80%程度であり，レイアウト設計の効率化を図るためにＤＭＳの図形探索機能を強化することの意義を再確認できた．

本稿ではソフトウェアによって実装されたデータ構造を評価したが，我々は連想メモリ（ＣＡＭ）を用いた図形処理プロセッサによってＤＭＳ関数を実装した場合についても同様の評価を行っている．[9]

謝辞

配線アプリケーションに関して有益なご助言をいただき，計算機実験についてもご協力下さった本大学大学院石川拓也氏に感謝いたします．

参考文献

[1] 小島，大附，佐藤：レイアウト・データ管理方式に関する一考察，信学技報，VLD89-90，pp.51-58 (1989)．

[2] A.Pitakasanonkul et al.:Comparisons of Quad Trees and 4-D Trees:New Results, IEEE Trans. on Computer-Aided Design, vol.8, No.22, (1989).

[3] R.A.Finkel and J.L.Bentley: Quad-trees--A data structure for retrieval on composite keys, Acta Inform.,vol.4,no.1-9, (1974).

[4] 乗松，大附，佐藤：Quad Trees に対する比較実験と考察，信学技報，COMP90-34, pp.35-44 (1990).

[5] G.Suzuki: Practical Data structure for Incremental DRC and Compation, Proc.of ICCD '87, pp.442-447 (1987).

[6] E.M.McCreight: Priority Search Trees, SIAM J. Comput..,Vol.14,No.2, pp.257-276 (1985).

[7] 池田　他：レイアウトデータ管理システムに対する各種データ構造の比較，情処研報，DA58-2, (1991).

[8] 石川　他：改良線分展開法の多層化，信学技報，VLD91-85, pp.41-48 (1991).

[9] 久保田　他：連想プロセッサを用いたレイアウト・データ管理システムの高速化，情処研報，DA57-1, (1991).