VLSI マスクパターン・オンライン設計規則チェック・システム†

鈴木五郎†† 薄井勝夫††† 岡村芳雄††††

VLSI マスクパターンを編集する過程で同時進行的に設計規則チェックを行うシステムを開発した．本システムは 3.5MIPS のワークステーション上で稼働しており，1) 46種の基本演算子を組み合わせて記述された設計規則を処理することにより，複雑で多岐にわたるチェックが可能，2) 2〜3 個のパターンであれば平均1秒以内のコマンド応答で設計規則チェック付き入力，コピー，移動，回転等が行える，3) 最も単純なデータ構造と比べてデータ量のオーバヘッドが10%程度，等の特長を持つ．特に2)，3)を実現するために，フィールドブロックと呼ぶ新しいデータ構造を開発した．メモリやマイコンのセル設計に適用した結果，本システムは有効で実用的であることが確認できた．

1. まえがき

VLSI マスクパターン設計では従来バッチ処理的に設計規則チェックが行われてきた．つまりマスクパターンの編集が終了した時点で，図面全体を一挙にチェックする方式である．ところが設計効率を上げたいことから，最近ではマスクパターンを編集する，つまり入力，コピー，移動，回転，削除等の操作をする過程で同時進行的にオンラインで設計規則チェックを行いたいというニーズが増加してきている．このようなオンライン設計規則チェック・システムは，1) 複雑で多岐にわたる設計規則が扱える，2) 設計規則チェック付きのマスクパターン編集が，チェックなしの純粋なパターン編集並みに高速に行える，3) 最も単純なデータ構造と比べてデータ量のオーバヘッドが少ない，等の条件を具備しなければならない．

オンライン設計規則チェック・システムとしては，MAGIC[1),2)]，SDA[3)]，EMILIE 2[4)]，CASSIOPEE[5)] 等が有名であるが，これらのシステムでは特に上記条件2) の高速処理を実現するため，タイトル[6)]や4分木[7)]と呼ぶ独自のデータ構造を持っている．しかし我々の評価[8)]では，タイルデータ構造は構造が複雑であるためにその更新処理に時間がかかり，かつデータ量が膨大で実用的でない，また4分木データ構造はプログラム・インプリメントが難しい，といった問題点があることを確認している．そこで今回これらの問題点を解決すべく，新たにフィールドブロックと呼ぶ独自のデータ構造を開発した．

本論文では，まず複雑で多岐にわたる設計規則を我々のシステムではどのようにして扱うかを述べ，次にフィールドブロックデータ構造について説明し，最後にシステムの評価結果を示す．

2. 設計規則の表現方法

典型的な設計規則の例を図1に示す．これは，PS (polysilicon) 層と CONT (contact) 層パターン間の間隔を規定するものであるが，チェックしたいパターン群の背景に，ある層のパターンが存在する場合としない場合で，間隔を変えなければならない規則となっている．つまり，図1の左の場合は右と異なり，背景に N (N type bury) 層のパターンが存在することから，設計規則 S2 は設計規則 S1 と異なるものとなる．

この例のような設計規則を計算機にどのように教え込むかがまず問題となる．設計規則の表現方法，特に条件の表現方法としては，1) 条件ごとに，それを記述する言語を用意する，2) 何種類かの基本的な図形演算子を用意しておき，それらを組み合わせて条件を表現する，等があるが，将来出現するであろう種々の条件にも柔軟に対応できるように，我々は 2) の方法を採用することにした．例えば上記設計規則は3つの基本演算子 AND, SUB, SPACE の組み合わせで記述されることになる．図1の最初の2行で，N層パターン内部に存在する D (diffusion) 層パターンと外部にあるD層パターンを分離する．次の2行でゲートの部分だけを取り出し，最後の2行でゲートと CONT 層パターン間の間隔を計算して，それぞれ1.5と1.0ミ

† Online Design Rule Checking System for VLSI Mask Pattern Design by GORO SUZUKI (Hitachi Research Lab., Hitachi Ltd.), KATSUO USUI (Hitachi Engineering Co.) and YOSHIO OKAMULA (Device Development Center Hitachi Ltd.).

†† (株)日立製作所日立研究所
††† 日立エンジニアリング(株)
†††† (株)日立製作所デバイス開発センタ

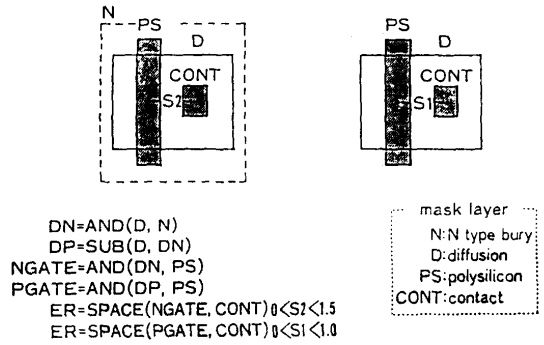
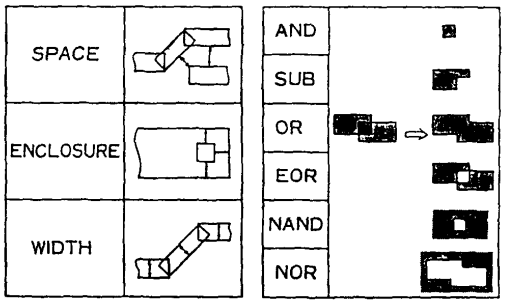


図 1 設計規則例
Fig. 1 Design rule example.



(1) チェック演算子 (2) ブール演算子
(1) Checking operators. (2) Boolean operators.

図 2 基本演算子
Fig. 2 Basic pattern operators example.

クロン以下ならばエラー層にエラー情報を出力する．本システムではチェック演算子，ブール演算子，それにトポロジ演算子の3つのグループに分けられる合計46種の基本演算子（**図2**にその代表例を示す）を用意しており，これらの基本演算子を組み合わせることによって複雑で多岐にわたる設計規則の表現が可能となった．

3. フィールドブロックデータ構造

3.1 新データ構造開発の背景

図1で説明した設計規則をチェックしながらマスクパターンを入力した場合の計算機処理手順を考えてみる．ここでは，**図3**のようにPS層のパターンを入力したとすると，

（P1） チェックしたい設計規則に関係する層のパターンで，入力パターンの近傍に存在するものを抽出する．

（P2） 抽出されたパターンと入力パターンに注目して，図1で説明した基本演算子列を上から順に実行する．

この一連の処理で，近傍パターンを検索する（P1）の処理が全体の処理時間を大きく支配することが経験上分かっている[8]．そこで，全体を高速処理するには近傍パターンを高速検索できるデータ構造が必要となる．第1章で述べたタイルデータ構造を評価した結果[9),10)]では，5～6個程度の少量のマスクパターンを設計規則チェック付きで編集する場合はチェックなしの純粋なパターン編集並みの速度で処理できるものの，50個程度の大量なパターンを編集する場合には処理速度が大きく劣化（純粋なパターン編集と比較して

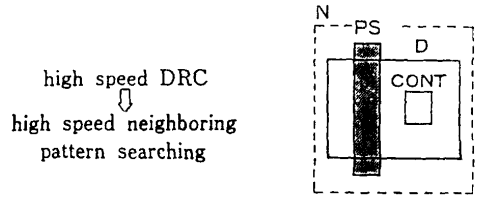


図 3 新データ構造開発の背景
Fig. 3 Motivation of the novel data structure development.

20～30倍程度の時間）してしまうことが分かった．また，最も単純なデータ構造と比べてデータ量が5～6倍になってしまう．今回我々が提案するフィールドブロックデータ構造はこのような問題点を完全に解決することができる．

3.2 フィールドブロックデータ構造の基本概念

本データ構造の基本概念を**図4**に示す．データ構造の中心は，メモリプレーンと呼ぶ2次元のメモリである．このメモリプレーンはいわばメモリのタンスに例えられ，同じ大きさを持つメモリの引き出しを複数個持っている．例えば対話型のパターン編集システムを

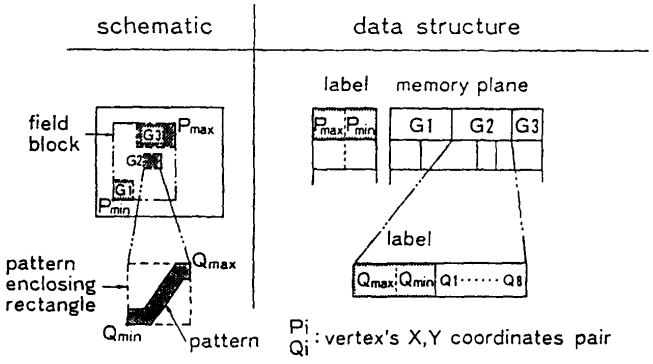


図 4 フィールドブロックデータ構造の基本概念
Fig. 4 Field block data structure.

用いてマスクパターンを入力する場合，主にパターンの頂点 X, Y 座標から構成されるパターンデータが，メモリの引き出しへ入力順に格納される．図4で，G1, G2, G3 の3つのパターンをこの順に入力した場合，それらのデータはメモリタンスの一番最初の引き出しに格納される．最初の引き出しは，3つのパターンデータで満杯になるため，次に入力されたパターンのデータは次の引き出しに格納される．G1, G2, G3 の3つのパターンは図面上である領域を占有しているが，これら3つのパターンを完全に包含する面積最小の矩形をもって占有領域を定義することにした．この占有領域をフィールドブロックと呼ぶ．つまり，各メモリの引き出しに格納されているパターンを完全に包含する面積最小の矩形としてフィールドブロックは定義される．各メモリの引き出しにはラベルを持たせており，フィールドブロックの左下と右上の頂点 X, Y 座標情報を格納しておく．この例では P_{min} と P_{max} の X, Y 座標が格納されている．また，各パターンデータにもパターン包含矩形の左下と右上の頂点 X, Y 座標情報を持つラベルを持たせてある．ここでパターン包含矩形は，該当パターンを包含する面積最小な矩形として定義される．この例では Q_{min} と Q_{max} の X, Y 座標がラベルに格納される．

このように，フィールドブロックデータ構造はラベルを階層的に持つことによって，パターンの存在位置を階層的に，しかもマクロに把握できる構造になっている．

3.3 データ構造の更新

マスクパターンを編集する過程でフィールドブロックデータ構造をどのように更新するかを説明する．**図5** では G1 と G2 の2つのパターンから構成される1個のフィールドブロックが存在しているが，ここに新しいパターン G3 を入力した場合，フィールドブロックの形状は左から右に変化する．この過程での主な更新処理は，該当するメモリの引き出しが持つラベルを左から右へ変更する作業である．仮にフィールドブロックの完全に内側（パターン包含矩形がフィールドブロックに完全に包含される位置）にパターン G3 を入力した場合，ラベル更新処理は必要なくなる．このように，フィールドブロックデータ構造はデータ構造の更新が極めて容易である特長を持っている．

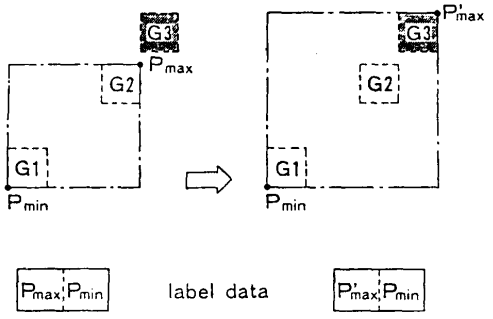


図 5 データ構造の更新
Fig. 5 Data structure maintenance.

3.4 近傍パターン検索アルゴリズム

次にオンライン設計規則チェック手順に沿って近傍パターンを検索するアルゴリズムを説明する．

図6 ではパターン間の間隔をチェックしながらハッチングをかけたパターンを入力している．

（S 1） 入力パターンのパターン包含矩形を設計規則の最大値 D だけ拡大する．

（S 2） 拡大された入力パターン包含矩形と共通領域を持つフィールドブロックを選択する．この例では，フィールドブロック FB1 が選択される．

（S 3） 拡大された入力パターン包含矩形と共通領域を持つパターン包含矩形を（S2）で選択したフィールドブロック中から選択する．この例では，パターン包含矩形 G3 が選択される．

（S 4） 必要ならば，（S3）で選択されたパターン包含矩形を持つパターンと入力パターンの間でブール演算あるいは／およびトポロジ演算を行う．

（S 5） （S3）で選択されたパターンあるいは（S4）

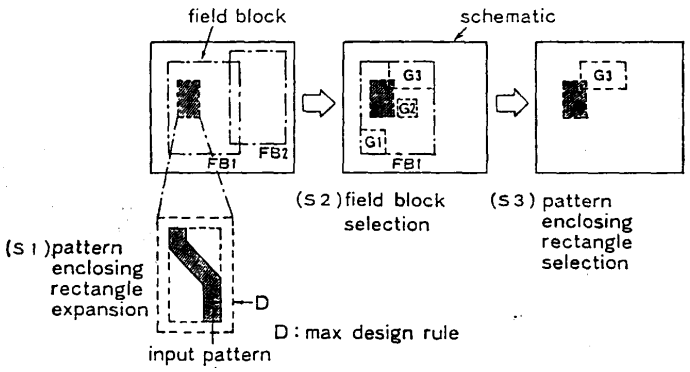


図 6 オンライン DRC 処理手順
Fig. 6 Online DRC process.

で演算した結果と入力パターンの間でチェック演算を行う．この例題ではパターン間の距離を計算し，入力パターンが設計規則を満足するか否かをチェックする．ここで2つのパターン間の距離は，以下の要領で計算する．パターン包含矩形どうしの共通領域に存在する辺の外側に向けて**図7**のように幅 D ((S1) と同一の値) の矩形を設定し，その矩形にかかる相手パターンの辺に注目し，2辺の4つの端点から相手の辺までの4つの距離（端点から相手パターンの辺に垂線が下ろせる場合には垂線の長さ．垂線が下ろせない場合には近い端点までの距離）①〜④を計算する．

上記手順において (S2) と (S3) が近傍パターンを検索する処理であるが，基本的には2つの矩形が共通領域を持つか否かを階層的にチェックするものであるから高速に処理することができる．

3.5 近傍パターン検索の高速化

近傍パターン検索をさらに高速化するため，主に次に示す2つのアイデアを採用することにした．

第1のアイデアはフィールドブロックそのものの階層化である．**図8**の例ではマクロフィールドブロックを導入している．このマクロフィールドブロックはその中に複数個のフィールドブロックを含んで構成される．データ構造としては第3.2節で説明した概念をそのまま素直に拡張すれば対応できる．つまり，パターンデータの代わりにフィールドブロックの形状データが格納されるメモリの引き出しを別途設け，その中に格納されるフィールドブロックを完全に包含する面積最小の矩形つまりマクロフィールドブロックの左下と右上の頂点 X, Y 座標情報を格納したラベルを持たせれば良い．

第2のアイデアはフィールドブロックどうしの重なり制御である．**図9**では2個のフィールドブロックが存在するが，パターン G3 を(1)のように移動した場合はフィールドブロックの形状が(2)のように変化する．2個のフィールドブロックが重なる領域に新しいパターンを設計規則チェック付きで入力すると，2個のフィールドブロックを調べなければならず時間のかかる処理になる．このように，図面中のすべてのフィールドブロックが重なる領域にパターンを入力した場合は，特別のデータ構造を持つ意味がなくなってしまう．つまり，フィールドブロックどうしの重なりができるだけ少なくなるように制御したい．そこでメッシュの概念を導入した．**図10**のように図面をあらかじめ同一サイズのメッシュに分割しておき，各メッシュに対応して第3.2節で説明したメモリの引き出しを用意しておく．そして，パターン包含矩形の左下頂点が存在するメッシュに対応するメモリの引き出しに該当パターンデータを格納する．図10の左の例では，パターン G1, G2, G3 のパターン包含矩形の左下頂点は i 番目のメッシュに存在し，パターン G4, G5 のパターン包含矩形の左下頂点は j 番目のメッシュに存在するため，G1, G2, G3 のパター

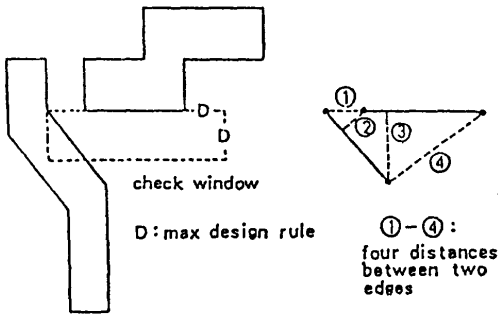


図7 パターン間距離計算
Fig. 7 Distance calculation between two patterns.

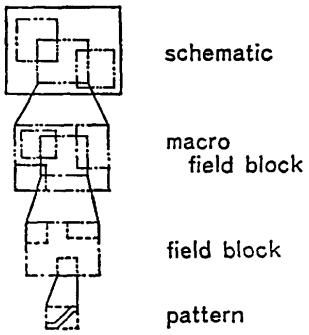


図8 階層フィールドブロック
Fig. 8 Hierarchical field block.

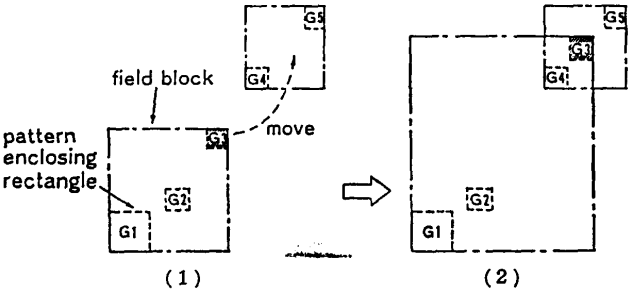


図9 フィールドブロック重なり制御 (1)
Fig. 9 Field block size control (1).

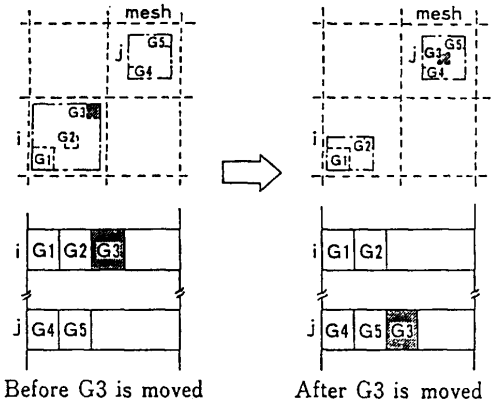


図 10 フィールドブロック重なり制御（2）
Fig. 10 Field block size control (2).

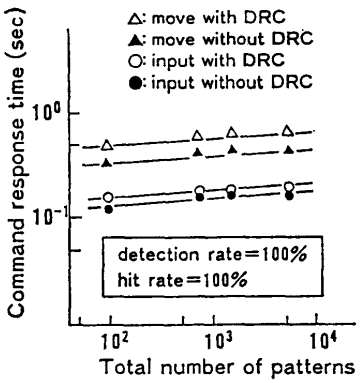


図 11 パターン編集とオンライン DRC 性能評価
Fig. 11 Online DRC and pattern editing performance.

ンデータと G4, G5 のパターンデータはそれぞれ i 番目と j 番目のメッシュに対応するメモリの引き出しに格納される.

ここで例えばパターン G3 を右上に移動した場合（図 9 と同じ例題），パターン G3 のデータは i 番目のメモリ引き出しから j 番目のメモリ引き出しに移動しておく．理由は移動後のパターン G3 包含矩形の左下頂点が j 番目のメッシュに存在するからである．パターン G3 移動後も，図 9 の例題と異なり 2 つのフィールドブロックは重ならなくなる．このように，メッシュの概念を導入することで，フィールドブロックどうしがなるべく重ならないように制御することが可能となる.

もちろん，複数のメッシュにまたがるような位置にパターンを配置あるいは移動したような場合は，フィールドブロックは複数のメッシュを覆うように広がる．その結果として，フィールドブロックどうしが重なる確率が高くなるが，本データ構造の思想をあまり複雑にしないために特別な対策は施してはいない．メッシュの概念を導入しているが，あくまでもフィールドブロックの広がりを制御するためであり，近傍パターンの検索は第 3.4 節で述べたアルゴリズムを使用している.

4. システムの評価結果

このオンライン設計規則チェック・システムを約 3.5 MIPS のワークステーション上で動いている対話型マスクパターン設計システム SPACE[11),12)] に組み込み，メモリ LSI 周辺回路セルのマスクパターン設計に適用して性能評価を行った．グラフィック・インタフェースとしては GKS[13)] を使用している.

第 1 に設計規則チェック付きのマスクパターン編集とチェックなしの純粋なパターン編集の速度を評価した．規模の異なる図面，つまりいろいろなパターン数を持つ図面で 3 個のパターンを入力あるいは移動（5 項目の設計規則チェック付きとチェックなし）した場合のコマンドの応答時間を測定した．**図 11** に評価結果を示す．X 軸は図面に含まれるパターンの数を，また Y 軸はコマンドの応答時間を表している．●はチェックなしでパターンを入力した場合，○はチェック付きでパターンを入力した場合，▲はチェックなしでパターンを移動した場合，また△はチェック付きでパターンを移動した場合をそれぞれあらわしている．すべてのコマンド応答時間は 1 秒以内であり，純粋な編集だけでなく設計規則チェック付きのマスクパターン編集も高速に行えることが分かる．上記したすべての場合，エラーの検出率とヒット率の両方が 100% である．ここで検出率 100% は設計規則の違反個所をすべて見逃さずに指摘したことを意味し，ヒット率 100% は指摘したエラーはすべて真のエラーであること（擬似エラーの指摘を完全に排除）を意味する．第 2 章で述べたように 46 種の基本演算子の組合せでかなり複雑な設計規則も表現できることから，検出率およびヒット率を 100% にすることが可能になった.

第 2 に 1 個のフィールドブロック当たり平均何個のフィールドブロックと重なるかを 2000 個のトランジスタを持つセルを用いて評価した．設計規則チェックでは全マスクパターン層を同時にチェックする必要はなく，通常 1 個の設計規則当たり 2〜3 のマスクパターン層だけをチェックすれば良い．そのため，実際には第 3.2 節で説明したメモリプレーンをマスクパターン層ごとに用意し，不要な層のデータは始めから

表 1 フィールドブロック重なり評価
Table 1 Average number of overlaps per one field block.

Mask layer	Polysilicon	Diffusion	Contact	Metal 1	Metal 2
Average number of overlaps (K)	0.5	0	0	0.4	1.7
Number of field blocks	88	44	78	10	48

参照しないで済むようにしている．そこで，マスクパターン層ごとに1個のフィールドブロック当たり平均何個のフィールドブロックと重なるかを測定した．重なり平均個数 K は次式で定義している．

$$K=\sum_{i=0}^{\infty} iN(i) \Big/ \sum_{i=0}^{\infty} N(i)$$

$N(i)$：i 個のパターンと重なるパターン数

表 1 に評価結果を示す．拡散層とコンタクト層では，$K=0$ である．一番大きな値でもメタル2層で $K=1.7$ であり，第3.5節で説明した第2のアイデアによってフィールドブロックの重なりがうまく制御できていることが分かる．

第3はタイルデータ構造とフィールドブロックデータ構造の性能比較である．オリジナルなタイルデータ構造では斜めの辺を持つパターンは扱えないことからデータ構造の拡張を行った．**図 12** にその概念を示す．この図面には2個のパターンが存在するが，パターンの外側と内側を台形状のタイルに分割し，各タイルには辺ごとにその辺に隣接するタイルの1つを指し示す隣接ポインタを持たせる．この隣接ポインタをたどることによって近傍パターンを容易に検索することができる．例えば右側のパターンから出発して，隣接ポインタを破線のようにたどり左側のパターンを探しあてることができる．2つのデータ構造をプログラム化し，2000トランジスタから構成されるセルで性能を評価した．性能比較を**表 2** に示す．フィールドブロックデータ構造は拡張タイルデータ構造の 20% のデータ量であり，最も単純なデータ構造と比べてもデータ量のオーバヘッドが 10% 程度である．50 個のパターンをコピーする純粋なパターン編集コマンドの応答時間は拡張タイルデータ構造の 5%，1項目の設計規則のチェックをかけながら 50 個のパターンをコピーするコマンドの応答時間に関しては 50% であることが分かった（ここで，応答時間はコピーするパターン数に比例しており，設計規則チェックをかけながら数百個程度の大量のパターンをコピーする場合でも 2〜3分程度の応答時間である）．以上からフィールドブロックデータ構造は拡張タイルデータ構造よりも高性能だと言える．拡張タイルデータ構造は構造が複雑な分だけデータ量が多くなり，またデータ構造を更新する処理に時間がかかる．一方構造の簡単さがフィールドブロックデータ構造を高性能にしている．

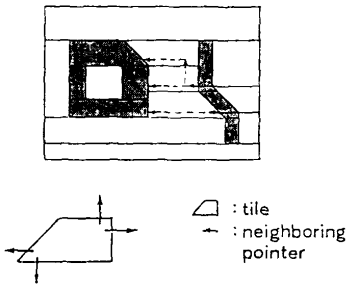


図 12 拡張タイルデータ構造
Fig. 12 Extended tile data structure.

表 2 データ構造の性能比較
Table 2 Comparison between two data structures.

Items \ Data structure	Extended tile data structure	Field block data structure
memory space	695 Kbyte	139 Kbyte
50 patterns copy	30 sec	1.5 sec
50 patterns copy with DRC	35 sec	16 sec

5. むすび

次の3つの特長を持つ VLSI マスクパターン・オンライン設計規則チェック・システムを開発した．

1) 複雑で多岐にわたる設計規則のチェックが可能．
2) 2〜3個のパターンであれば平均1秒以内のコマンド応答で設計規則チェック付きパターン入力，コピー，移動，回転等が行える．
3) 最も単純なデータ構造と比べてほとんどオーバヘッドのないデータ量．

現在メモリやマイコンのセル設計に本システムを適用しており，設計工数低減に大きく寄与している．

謝辞 本研究の機会を与えていただいた当社平沢主管技士長，前島部長，細坂部長はじめ関係諸氏に深く感謝します．

参考文献

1) Ousterhout, J.K. et al.: MAGIC: A VLSI

Layout System, *Proc. of 21st Design Automation Conference*, pp. 152-159 (1984).
2) Taylor, G.S. et al.: MAGIC's Incremental Design Rule Checker, *Proc. of 21st Design Automation Conference*, pp. 160-165 (1984).
3) Swartz, P.A. et al.: Incremental Design Rule Checking, *Proc. of Custom Integrated Circuits Conference*, pp. 60-63 (1985).
4) Baraban, L. et al.: An Interactive Design Rule Checker, *Proc. of International Conference on Computer Aided Design*, pp. 213-214 (1983).
5) Berger, J. et al.: A Range Searching Subsystem Used to Perform Efficient VLSI Design Check, *Proc. of International Conference on Computer Aided Design*, pp. 408-411 (1986).
6) Ousterhout, J.K.: Corner Stitching: A Data Structure for VLSI Layout Tools, *IEEE Trans. on Computer Aided Design*, pp. 87-100 (1984).
7) Kedem, G.: The Quad-CIF Tree: A Data Structure for Hierarchical Online Algorithms, *Proc. of 19th Design Automation Conference*, pp. 352-357 (1982).
8) Suzuki, G.: Practical Data Structure for Incremental Design Rule Checking and Compaction, *Proc. of International Conference on Computer Design*, pp. 442-447 (1987).
9) 鈴木五郎：オンライン設計用タイル型データ構造の基礎検討，情報処理学会論文誌，Vol. 30, No. 4, pp. 518-526 (1989).
10) 鈴木五郎：オンライン設計用タイル型データ構造の検討，昭和 62 年度信学部門全大，p. 102 (1987. 11).
11) 鈴木五郎ほか：ASIC 用対話型セル設計システム SPACE，情報処理学会設計自動化研究会，46-7, pp. 49-55 (1989. 2).
12) 加藤真司ほか：SPACE：対話型マスクパターン設計システム，情報処理学会全国大会論文集，p. 1800 (1988. 9).
13) Hopgood, F.R.A. et al.: *Introduction to the Graphical Kernel System*, Academic Press, London (1983).

(平成 3 年 1 月 17 日受付)
(平成 3 年 5 月 7 日採録)

鈴木 五郎（正会員）

昭和 50 年慶応義塾大学理工学部電気工学科卒業．同年(株)日立製作所入社．以来 VLSI・DA/CAD システムの研究開発に従事．計算幾何学に興味を持ち，レイアウト・コンパクタ，設計規則検証，回路接続検証，図面エディタ，モジュール・ジェネレータを開発．現在同社日立研究所第 8 部主任研究員．IEEE，電子情報通信学会各会員．

薄井 勝夫（正会員）

昭和 19 年生．昭和 37 年日立工業専修学校機械科修了．同年(株)日立製作所入社．計算機制御システム，ソフトウェア工学，CAD システムの研究開発を経て，昭和 60 年より日立エンジニアリング(株)に勤務．現在，VLSI の CAD システムの開発に従事．

岡村 芳雄

昭和 50 年東京工業大学工学部電子工学科卒業．同 52 年，同学科修士修了．同年(株)日立製作所入所．以来 LSI マスクパターンレイアウト，検証用 DA/CAD システムの開発に従事．マスクパターンからの回路抽出と特性検証，マスクパターンデータベース・マスクパターンエディタを開発．現在同社デバイス開発センタ DA 開発部主任技師．