DRC ルールファイルからの設計規則抽出とその可視化

北浦直樹† 越智裕之† 津田孝夫†

レイアウトの設計規則検査（DRC：Design Rule Check）とは，レイアウトのパターンが当該プロセスの設計規則に違反していないかどうかを検証することである．DRC ツールを使うためには設計規則をツール固有の書式で表したルールファイルが必要であるが，これは通常，書面の設計規則書に基づき人手で作成される．LSI の設計・製造に万全を期するためには，人手で作成されたルールファイルを何らかの方法で検証することが強く望まれるが，ルールファイル自体を検証する方法は従来ほとんど研究されていない．本論文ではルールファイルと設計規則書の照合を支援するため，ルールファイルから設計規則を抽出し，可視化するための手法について考える．これまでに我々は命令列のパターンマッチングに基づく方法を提案しているが，ルールファイルは同じ設計規則違反を報告するものであっても命令の組み合わせ方や順序に任意性があるため，たとえば 70 個の設計規則を持つプロセスのルールファイルに対応させるため，43 種類ものテンプレートが必要であった．本論文では，設計規則違反を報告する命令に対応づけられた表示用テンプレートと，その命令で参照されるレイヤの表示アルゴリズムを独立させることにより，この問題点を改善した．提案した手法に基づき，Cadence 社 Dracula 用の DRC ルールファイルから設計規則を抽出，可視化するツールを開発し，これに 70 個の設計規則を持つプロセスのルールファイルを適用したところ，わずか 6 種類のテンプレートにより 65 個の設計規則について設計規則書とほぼ同様の表示が得られた．

Design Rule Extraction from DRC Rule Files, and Its Visualization

Naoki Kitaura,† Hiroyuki Ochi† and Takao Tsuda†

DRC (Design Rule Check) is a procedure to verify whether a layout pattern violates the design rules of the process. In order to use a DRC tool, a tool-specific rule file which describes the design rule is required. The rule file is usually created manually, based on the design rule given by documents. To ensure perfect design and fabrication of LSI, it is strongly desired to verify the manually-created rule file. However, few studies have been made for the verification methodology of the rule file itself. In this paper, we will consider to extract and visualize the design rule from DRC rule file in order to compare it with original documents. We proposed a method based on pattern matching of command sequence. However, it needed 43 templates for a rule file of a process with 70 design rules, because there is no uniqueness in usage nor ordering of commands in a rule file to report a design rule violation. In this paper, we solve the problem by introducing a display template corresponding to a error-reporting command and, independently, a display algorithm for reference layers. Based on the proposed method, we have developed a tool to extract and visualize design rules from DRC rule file for Cadence Dracula. It displayed 65 out of 70 design rules of a process using only 6 templates.

1. はじめに

集積回路設計で使用される自動化ツールの 1 つにレイアウトの設計規則検査（DRC：Design Rule Check）を行うものがある．ここで DRC とは，レイアウトのパターンが当該プロセスの設計規則に違反していないかどうかを検証することである．

DRC を実行するためには，ルールファイルと呼ばれるものが必要となる．これは対象とするプロセスの設計規則を DRC ツールが解釈できる形式で記述したテキストファイルであり，多くの DRC ツールのルールファイルは，設計規則違反個所を取り出すためのパターン演算の命令列として記述される．このルールファイルは設計規則を正確に反映していなければならないが，設計規則を記述するための数学的に厳密な裏付けのある書式はなく，設計規則は通常，英語あるいは日本語および図面で記された書面（設計規則書）で与えられる．このため，設計規則書に基づいてルールファイルを作成する作業は人手によって行わなければ

† 広島市立大学情報科学部情報工学科
Department of Computer Engineering, Faculty of Information Sciences, Hiroshima City University

ならない．プロセスの進歩にともないルールファイルがますます複雑長大になる中，LSIの設計・製造に万全を期するためには，人手によって作成されたルールファイルを何らかの方法で検証することが強く望まれるが，そのような検証を行うツールはこれまでほとんど存在しなかった．

本論文ではルールファイルと設計規則書を照合することを支援するため，DRCで使われるルールファイルの中の設計規則記述から設計規則を抽出し，これを可視化するための手法について考える．

パターン演算の命令列として記述するタイプのルールファイルから設計規則をリバースエンジニアリングするためには，ルールファイルの中から設計規則に対応する命令列を認識することが必要であるが，このような手法の研究は，我々の知る限り，これまで行われていない．一方，ソフトウェア分野においては手続き型プログラミング言語の最適化コンパイラにおいて，プログラム中のループの自動認識などの研究が古くから行われている[1]．特にスーパコンピュータ用の自動ベクトル化コンパイラにおいては，特定のマクロ命令などに置き換えられるループを検出するイディオム認識と呼ばれる技術が研究されている[2]．

本論文の対象とするルールファイルも，レイヤという型のデータに対してパターン演算を逐次的に施す一種の手続き型プログラミング言語と見なすことができる．一般に手続き型プログラミング言語は，同じ処理を行うものであっても記述に任意性があるので，そこから特定の意味内容を持つ命令列を抽出することは容易ではない．ルールファイルから設計規則を抽出するための手法として，これまでに我々は命令列のパターンマッチングに基づく方法を提案しているが[3]，このようなアプローチでは，たとえば70個の設計規則を持つプロセスのルールファイルに対応させるため，43種類ものテンプレートが必要であった．

本論文で提案する手法は，設計規則そのものの表示アルゴリズムと，その設計規則で参照されるレイヤの表示アルゴリズムが独立している点で文献3)と大きく異なる．前者は最終的に設計規則違反を報告する命令に対応づけられたテンプレートに基づいており，後者はあらかじめルールファイル全体を走査して抽出される，派生レイヤの定義や，ヒューリスティックに推定されるレイヤ間の包含関係に基づいている．

提案した手法に基づき，Cadence社Dracula用のDRCルールファイルから設計規則を抽出，可視化するツールを開発した．このツールにモトローラ社1.2 μmメタル2層CMOSプロセスのルールファイルを適用したところ，70個の設計規則のうち65個について，わずか6種類のテンプレートにより可視化することができた．

以下，2章では準備として設計規則検査ならびに従来の手法について述べ，3章で提案する手法を述べる．4章では提案する手法に基づき開発したツールの実装と結果，および考察を述べる．5章でまとめと今後の課題を述べる．

2. 準　　備

2.1 設計規則と設計規則書

一般に，LSIはガラスマスク上に描かれたパターンをシリコン基板上に積み重ねるプレーナ技術で製造されている[4]．レイアウトパターンはこのマスク上のパターンを作成するための図形データの集合であり，実際には各工程のマスクに対応づけられたレイヤごとに区別して取り扱われる．レイアウト設計規則（以下，設計規則と呼ぶ）は，各レイヤの図形の最小幅，最小間隔や，あるレイヤの図形と別のレイヤの図形との重なりなどを定めているものである．

設計規則書は，設計規則が記された書面である．**図1**に設計規則書の例を示す．ここにはアクティブレイヤに関する設計規則が記されており，右の図面はレイヤの位置関係や規則を図示したもので，e1〜e6が左の設計規則の説明と対応している．

e1	Minimum size	3um
e2	Minimum spacing	4um
e3	N-well overlap of p+	5um
e4	N-well overlap of n+	2um
e5	N-well space to n+	7um
e6	N-well space to p+	6um

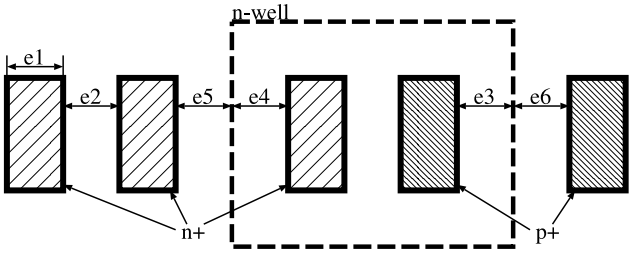


図1 設計規則書の例
Fig. 1 An example of design rule manual.

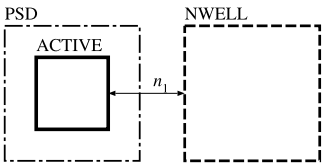
2.2 設計規則検査（DRC）とルールファイル

設計規則検査（DRC：Design Rule Check）とは，設計されたレイアウトのパターンが当該プロセスの設計規則に違反していないかどうかを検証することである．DRC は，DRC ツールを用いて計算機上で行われる．DRC を行うためには，設計規則を DRC ツールが解釈できる形式で書き表したルールファイルが必要である．

ルールファイルはテキスト形式で書かれており，その記述の形式はツールによって様々であるが，大きく 2 つのタイプに分類することができる．

1 つは，属性のキーワードとパラメータの組を列挙するものである．ここで属性とは「MET1 の最小幅」，「MET1 の最小間隔」などであり，あらかじめツール側で定義されたキーワードに対応づけられている．このタイプのルールファイルは，DRC ツールよりはむしろ配置配線ツールに多くみられる．記述に任意性がなく，ツールがルールファイルを読み込むことは容易である反面，テクノロジの進歩によって新しい規則が登場した場合には，ツールをバージョンアップして新たなキーワードを追加するなどの対応が必要となる欠点がある．

もう 1 つは多くのレイアウト検証専用ツールにみられるもので，パターン演算の命令列で設計規則を表すタイプのルールファイルである．つまり，レイアウト設計で使われるレイヤ（以下，ベースレイヤと呼ぶ）に対して種々のパターン演算[5]を適用し，設計規則違反箇所を取り出そうというものである．ここで，パターン演算とは各レイヤの図形領域どうしの論理演算，包含関係や重なり関係を調べる位相演算，図形どうしの接続の有無や，接続の仕方を調べる接合演算，図形の寸法を調べる計量演算，図形の拡大，縮小を行うリサイジングなどのことをいう．設計規則違反箇所を取り出すための命令列の中では，ベースレイヤとは別にパターン演算によって二次的に生成されるレイヤがある．このレイヤを派生レイヤと呼ぶ．例として設計規則とそれに対応する記述をそれぞれ，**図 2**，**図 3** に示す．ここで，NWELL と ACTIVE と PSD はベースレイヤとする．図 3 の 1 行目は，ACTIVE レイヤ図形から NWELL レイヤ図形と重なる領域を除いた図形を求め，これを派生レイヤ NDIF とする論理演算 not コマンドである．2 行目は，NDIF レイヤ図形と PSD レイヤ図形の重なった領域の図形を派生レイヤ PSUB とする論理演算 and コマンドである．3 行目は，PSUB レイヤ図形と NWELL レイヤ図形との間隔が n_1 未満であれば err01 というエラーを報告する計量演算 ext コマンドである．



Min Space (ACTIVE (in PSD)–NWELL) : n_1

図 2 設計規則 err01
Fig. 2 Design rule “err01”.

```
not   ACTIVE  NWELL   NDIF
and   NDIF    PSD     PSUB
ext   PSUB    NWELL   lt n1      output err01
```

図 3 設計規則 err01 の記述
Fig. 3 Description for design rule “err01”.

パターン演算の命令列で設計規則を表現する方法は，複雑な設計規則でも演算を組み合わせることによって記述することができる柔軟性がある．その一方，設計規則を表現するためのパターン演算の命令列は，レイヤという型のデータに対して逐次的に操作を行う，いわば手続き型のプログラミング言語であるから，同じ設計規則を表現する場合でも，そのパターン演算の組み合わせ方や順序には任意性があり，このことは，ルールファイルの可読性や保守性を大きく損ねている．

2.3 素朴な手法とその問題点

本論文では人手によって作成された DRC 用ルールファイルを検証するための 1 つのアプローチとして，ルールファイルの中の設計規則記述から設計規則を自動的に抽出し，これを可視化することについて考える．たとえば，図 3 の命令列を含むルールファイルが入力として与えられたとき，図 2 のような表示を自動的に出力させようというものである．これはルールファイルの検証そのものを完全に自動化するものではないが，グラフィック表示された設計規則を設計規則書と見比べることによってルールファイルを容易に検証（確認）できるようになり，ルールファイルの開発や保守に携わる人間の負担を大幅に軽減できると期待される．

我々は，すでに設計規則をルールファイルから抽出して可視化する一手法を提案している[3]．これは，1 つの設計規則を表す一連のパターン演算命令列をルールファイルの中から取り出し，これをあらかじめ用意されたテンプレートと照合し，一致したテンプレートに基づいて表示を行うというものである．また，表示をより的確に行うため，ルールファイルの中から包含関係，独立関係，レイヤサイズなどのレイヤ属性情報を抽出しておき，テンプレートの選択に利用する手法

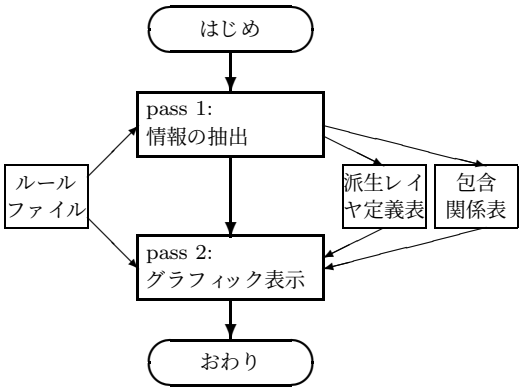


図 4 提案する手法の流れ図
Fig. 4 Flowchart of the proposed method.

も提案している．

この方法は，任意のパターン演算命令列に対してテンプレートを定義することができるので，原理的にはどのような複雑な設計規則にも対応することができる．しかし，同じ設計規則であっても命令列の記述方法には任意性があるため，膨大な量のテンプレートが必要であった．たとえば，70 個の規則を持つテクノロジのルールファイルに記述されているすべての設計規則を可視化するために 43 種類ものテンプレートが必要であった．

3. 提案する手法

提案する手法の目的は 2.3 節で述べた文献 3) の手法と同様，入力としてルールファイルが与えられたとき，設計規則をグラフィック表示することである．提案する手法の流れは**図 4** に示すとおり，情報を抽出する pass 1 とグラフィック表示を生成する pass 2 からなる．pass 1 では，各派生レイヤの定義，およびレイヤどうしの包含関係の情報を抽出する．pass 2 では，エラーを報告するコマンドに対応するテンプレートを検索し，そのテンプレートに基づき設計規則の表示を行う．ここでエラーを報告するコマンドの引数となっているレイヤを適切に可視化するため，pass 1 で抽出された情報を援用する．

提案手法が文献 3) と大きく異なるのは，pass 2 において，設計規則そのものの表示アルゴリズムと，その設計規則で参照されるレイヤの表示アルゴリズムを独立させた点である．これにより，少ないテンプレート数でより多くのルールファイル記述に対応することを狙っている．

例として図 3 の命令列を含むルールファイルが入力として与えられた場合について説明する．まず pass 1 で，1〜2 行目およびルールファイルの他の部分を走査した結果，以下の情報が得られる（詳しくは後述）．

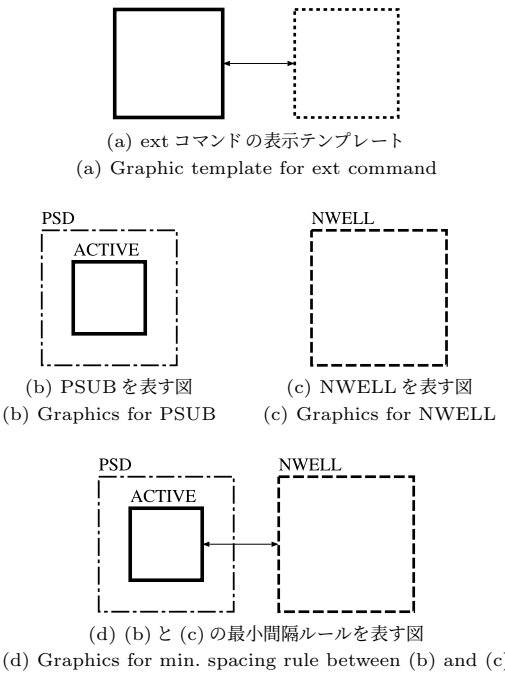
(1) （派生レイヤの定義）派生レイヤ PSUB はベースレイヤ PSD および ACTIVE の重なった領域である．
(2) （レイヤどうしの包含関係）レイヤ ACTIVE はレイヤ PSD に包含される．

次に pass 2 ではルールファイルを走査し，エラーを報告するコマンド☆を探す．この例では，3 行目の ext コマンドがこれにあたる．ext コマンドに対応する表示テンプレートは**図 5** (a) のようなものがあらかじめ用意されている．この左右の正方形はそれぞれ ext コマンドで参照されているレイヤ PSUB および NWELL に対応するので，この部分に PSUB および NWELL の表示をあてはめればよい．pass 1 で得られた上記の情報を基に派生レイヤ PSUB を表すと図 5 (b) のようになる．一方，NWELL レイヤはベースレイヤなので，図 5 (c) のように表せばよい．これら 3 つより，図 5 (d) のような表示が生成される．

次節から，それぞれについて詳しく述べる．

3.1 派生レイヤの定義の抽出

ルールファイルではしばしば派生レイヤが使われる



(a) ext コマンドの表示テンプレート
(a) Graphic template for ext command
(b) PSUB を表す図
(b) Graphics for PSUB
(c) NWELL を表す図
(c) Graphics for NWELL
(d) (b) と (c) の最小間隔ルールを表す図
(d) Graphics for min. spacing rule between (b) and (c)

図 5 図 3 から図 2 を生成する流れ
Fig. 5 Process for generating Fig. 2 from Fig. 3.

☆ Cadence 社 Dracula のルールファイルの場合，“output” が付されたコマンドがこれにあたる．

```
and   ACTIVE  NWELL  PDIF
not   ACTIVE  NWELL  NDIF
not   PDIF    PSD    NSUB
and   NDIF    PSD    PSUB
```

ただし，ACTIVE，NWELL，PSD はベースレイヤ

図 6 派生レイヤの定義の抽出の例
Fig. 6 An example for extraction of definition of derived layers.

表 1 派生レイヤ定義表
Table 1 Derived-layer definition table.

派生レイヤ＼ベースレイヤ	NWELL		ACTIVE		PSD	
	in	out of	in	out of	in	out of
PDIF	○		○			
NDIF		○	○			
NSUB	○		○			○
PSUB		○	○		○	

が，設計規則を表示する際にはベースレイヤだけを用いて表さなければならない．そこで，各派生レイヤがどのようなベースレイヤの組合せとして定義されているかの情報を抽出しておく必要がある．

提案する手法では派生レイヤの定義を保持するため，ベースレイヤと重なる（in），重ならない（out of）の 2 つのフラグを各派生レイヤに用意する．そして pass 1 でルールファイル中の命令列を走査している際に and コマンドまたは not コマンドに遭遇したとき，各派生レイヤに対してこれらのフラグを設定していく．

以下，**図 6** の例で考える．1 行目は，and コマンドを使った記述である．これより，PDIF レイヤ図形は ACITVE レイヤ図形と NWELL レイヤ図形の両方の重なりとして定義されていることが分かる．2 行目は，not コマンドを使った記述である．これより，NDIF レイヤは ACITVE レイヤ図形が存在する領域と NWELL レイヤ図形が存在しない領域の重なりとして定義されていることが分かる．3 行目，4 行目も同様であるが，参照している派生レイヤの属性が継承されることに注意されたい．これらより**表 1** に示すような属性が抽出される．

3.2 包含関係の抽出

レイヤ B の図形があり，その図形の内側にレイヤ A（の輪郭の全部または一部）が存在するとき，レイヤ B はレイヤ A を包含（enclose）していると呼ぶことにする☆．

派生レイヤを適切に可視化するためには，当該派生レイヤの元となっているベースレイヤどうしの包含関係に関する情報が必要である（この情報がどのように使われるかについては 3.3.1 項で述べる）．提案する手法では，レイヤどうしの包含関係を推定するヒューリスティックとして下の 3 つを導入する．

(1) enc A B lt n output err というコマンドがあれば B は A を包含する．
(2) select B enclose A C というコマンドがあれば B は A を包含する．
(3) not A B C というコマンドがあれば B は A を包含する．

pass 1 でルールファイル中の命令列を走査している際にこれらのコマンドに遭遇したならば，推定された包含関係を包含関係表（後述）に記録する．

(1) の enc コマンドは，レイヤ B の輪郭からその内側にあるレイヤ A の輪郭までの距離を規定する記述であり，たとえば A がコンタクトレイヤ，B がそれを囲む導体レイヤである場合などにしばしば見受けられる．これより，レイヤ A の図形とレイヤ B の図形が重なる場合は，B の内側に A（の輪郭の全部または一部）が存在するということが推定できる．

(2) の select コマンドは，レイヤ B の図形の中からレイヤ A の 1 個の図形全体を内側に有するものをレイヤ C とするコマンドである．これは，たとえば**図 7** にあるように，ゲートポリシリコンレイヤの図形の中から第 2 ポリシリコンレイヤの図形と容量を形成している図形を選択するという場面にみられる記述である．このため，レイヤ A の図形とレイヤ B の図形が重なる場合は，B の図形が A の図形を包含するということをヒューリスティックに推定することにする．

(3) の not コマンドは，レイヤ A の図形からレイヤ B の図形と重ならないものをレイヤ C とする記述である．ここでは**図 8** に示すように，ACTIVE レイヤの中から NWELL 外に配置されているもの（NMOS トランジスタまたは GND の TAP となる部分）を選択する場合などにしばしばこのコマンドが使われることを念頭におき，レイヤ B の図形とレイヤ A の図形が重なる場合はレイヤ B がレイヤ A を包含するということをヒューリスティックに推定することにする．た

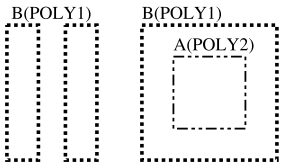


図 7 ヒューリスティック (2) の例
Fig. 7 An example for heuristic (2).

☆ 命題論理や集合論における包含（imply，subsume など）とは意味が違うことに注意されたい．

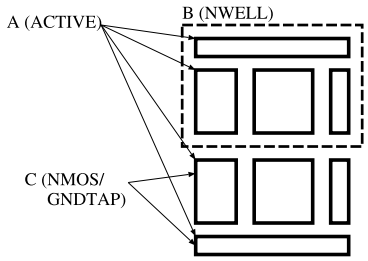


図 8 ヒューリスティック (3) の例
Fig. 8 An example for heuristic (3).

```
not   ACTIVE  NWELL   NDIF
enc   ACTIVE  PSD     lt n0     output err00
enc   POLY1   ACTIVE  lt n1     output err01
enc   ACTIVE  POLY1   lt n2     output err02
```

図 9 包含関係の抽出の例
Fig. 9 An example for extraction of enclosure relation.

表 2 包含関係表
Table 2 Enclosure relation table.

包含される	包含する
ACTIVE	NWELL
ACTIVE	PSD
POLY1	ACTIVE
ACTIVE	POLY1

だし，NOT コマンドはほかにも様々な場面で用いられるため，このヒューリスティックはレイヤ A，B ともにベースレイヤである場合に限って適用する．

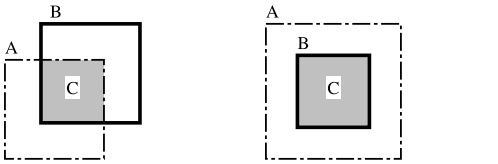
例として**図 9** の記述について考えると，1 行目の not コマンドにはヒューリスティック (3) が，2 行目以降の各 enc コマンドにはヒューリスティック (1) が適用されるので，**表 2** のような属性が抽出される．図 9 の 3 行目と 4 行目から抽出される属性は一見矛盾しているが，このようなルールファイル記述は MOS トランジスタのゲートポリシリコンと拡散層などでしばしば見受けられるので，この情報をこのまま pass 2 に渡すことにする．

3.3 設計規則の表示

3.3.1 派生レイヤの表示

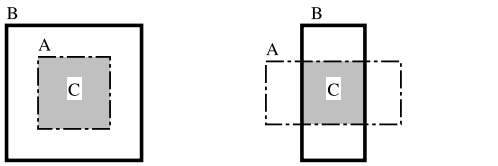
派生レイヤの表示は，その派生レイヤがどのようなベースレイヤで構成されるか（3.1 節）という情報とベースレイヤ間の包含関係（3.2 節）を使い行う．

以下提案する手法における派生レイヤの表示の基本的な流れを説明するため，コマンド “and A B C” によって 2 つのベースレイヤ A，B から生成される派生レイヤ C の表示方法を述べる．まず 3.1 節で述べたとおり，この and コマンドから，派生レイヤ C はベースレイヤ A および B の重なりとして定義されて



(a) A と B の包含関係が不明の場合
(a) When overlapping relation unknown

(b) A が B を包含する場合
(b) When A encloses B



(c) B が A を包含する場合
(c) When B encloses A

(d) A と B が相互に包含し得る場合
(d) When mutually enclosable

図 10 “and A B C” で得られたレイヤ C の表示
Fig. 10 Graphics for layer C derived from “and A B C”.

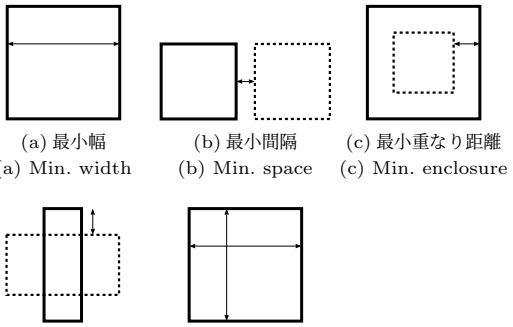
いるという情報が抽出される．このとき派生レイヤ C の表示は，ルールファイル全体を走査する中で 3.2 節で述べた方法により抽出されたベースレイヤ A および B の包含関係によって**図 10** に示す 4 種類のいずれかとなる．(a) は，ベースレイヤ A および B の包含関係がまったく推定できなかった場合に適用される最も一般的な表示である．(b) はレイヤ A がレイヤ B を包含するという包含関係のみが推定された場合に適用される表示であり，(c) はその逆の場合である．(d) は両方の包含関係が抽出された場合（図 9 の例の 3〜4 行目のような場合）に適用される表示であり，レイヤ A とレイヤ B が拡散層とゲートポリシリコンである場合などを想定したものである．

3 つ以上のベースレイヤによって派生レイヤが定義されている場合も，上に述べたアルゴリズムを繰り返し適用することにより表示を生成している．

3.3.2 設計規則の表示

多くの場合，ルールファイル中のエラーを報告するコマンドは設計規則書中の規則と 1 対 1 に対応づけられている．またエラーを報告するコマンドは，最小幅，最小間隔，重なり距離などを規定する計量演算のコマンドが用いられることが多く，その場合は対応する設計規則が容易に特定できる．提案する手法では，エラーを報告するコマンドとしてしばしば用いられるコマンドについて設計規則の図示方法をテンプレート化しておき，これを用いて設計規則を表示している．

以下，代表的なテンプレートについて述べる．



(a) 最小幅 (a) Min. width　(b) 最小間隔 (b) Min. space　(c) 最小重なり距離 (c) Min. enclosure

(d) 最小突き出し長さ (d) Min. extension　(e) コンタクトサイズ (e) Contact size

図 11 設計規則表示用テンプレートの例
Fig. 11 Example of graphics templates for design rules.

3.3.2.1 最小幅コマンド（width）のテンプレート

このコマンドは，ベースレイヤもしくは派生レイヤを 1 つ指定し，そのレイヤの最小幅を規定するものである．表示用のテンプレートを**図 11** (a) に示す．最小幅に関する設計規則の表示は，ベースレイヤもしくは派生レイヤに最小幅用のテンプレートを適用する．つまり，当該ベースレイヤまたは派生レイヤを表示し，その中の対象となるレイヤの内側に矢印を描くことによって行う．

3.3.2.2 最小間隔コマンド（ext）のテンプレート

このコマンドは，ベースレイヤもしくは派生レイヤを 2 つ指定し，その間の最小間隔を規定するものである．表示用のテンプレートを図 11 (b) に示す．最小間隔に関する設計規則の表示は，2 つのベースレイヤもしくは派生レイヤを左右に並べて表示し，その中の対象となるレイヤの外側どうしを矢印で結ぶことによって行う．

3.3.2.3 最小重なり距離コマンド（enc）のテンプレート

このコマンドは，ベースレイヤもしくは派生レイヤを 2 つ指定し，囲む側のレイヤと囲まれる側のレイヤとの距離を規定するものである．表示用のテンプレートを図 11 (c) に示す．最小重なり距離に関する設計規則の表示は width コマンドや ext コマンドとは違い，対象となる 2 つのレイヤ図形を合成して 1 つのレイヤ図形にまとめて表示する必要がある．提案する手法では，対象となる 2 つのレイヤに対し and コマンドを適用して合成される仮想的な派生レイヤを導入することにより，これを簡潔に実現している．

例として次の記述を用いて説明する．

```
enc      CONT    PSUB    lt n2       output err02
```

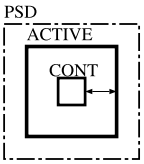


図 12 err02 の表示結果
Fig. 12 Generated graphics for “err02”.

これは PSUB レイヤに囲まれた CONT レイヤとの重なり距離が n_2 未満の場合 err02 となる記述である．ただし，派生レイヤ PSUB に関しては表 1 および表 2 の情報を用いるものとする．したがって，この設計規則を表す図はテンプレート上では PSUB レイヤ図形に囲まれた CONT レイヤ図形となるが，このうち PSUB レイヤは派生レイヤであるから，PSD レイヤに囲まれた ACTIVE レイヤとして表さなければならない．

提案するアルゴリズムでは，まず CONT レイヤ，PSUB レイヤのほかに err02 レイヤという派生レイヤを導入する．この err02 レイヤは CONT レイヤと PSUB レイヤの and 演算で得られるものと定義する．そして，この派生レイヤ err02 を 3.3.1 項で述べたアルゴリズムを用いて表示する．すると，包含関係表から得られる情報に従って外側から PSD レイヤ，ACTIVE レイヤ，CONT レイヤが順に重なり合う図形が生成される（**図 12**）ので，最後に enc コマンドの対象となる 2 つのレイヤの間に矢印を描くことにより表示が完成する．

3.3.2.4 最小突き出し長さを規定するコマンドのテンプレート

CMOS プロセスの設計規則書には必ず，図 11 (d) のような図が登場する．これはトランジスタを形成する際のゲートポリシリコンの最小突き出し長さ（end cap）などを説明するための図である．この場合もルールファイル上は enc コマンド☆によって設計規則違反が報告されるため，図 11 (c) のテンプレートが適用される場合と区別する必要がある．

図 11 (d) のテンプレートが適用されるべき状況で特徴的なのは，対象となる 2 つのレイヤ A，B に対し，A に対する B の最小突き出し長さを規定する enc コマンドと，B に対する A の最小突き出し長さを規定する enc コマンドの両方がルールファイル中に存在するという点である．たとえば MOS トランジスタの場

☆ ルールファイルによっては enc コマンドを含む数行の定型的な命令列が使われることもある．

合，チャネル領域からのゲートポリシリコンの最小突き出し長さだけでなく，チャネル領域からのソース・ドレイン領域の最小突き出し長さも規定されることが一般的である．

このことから,「A が B を包含する」ことと「B が A を包含する」ことの両方の情報が抽出された場合には，テンプレートとして図 11 (c) ではなく図 11 (d) を適用することとした．

3.3.2.5 コンタクトサイズを規定するコマンドのテンプレート

ほとんどのプロセスにおいて，コンタクトやビア（スルーホール）は所定のサイズの正方形以外を禁止している．これを規定するルールファイルの代表的な書き方は下のとおりである．このような 2 つのコマンドの組合せがあった場合に適用される表示用のテンプレートを図 11 (e) に示す．

```
width[L] CONT    seleq n    tmp1
not      CONT    tmp1       tmp2      output err03
```

3.3.2.6 その他の主なエラー報告コマンドに対する対応

レイヤ A とレイヤ B が重なることが禁止されている場合，エラーを報告するコマンドとして and コマンド（あるいは select コマンド）が用いられる．この場合，レイヤ A とレイヤ B が重なっている図に大きく×印を描画する．

ある特定のレイヤについて最小グリッドを規定する場合，snap コマンドと xor コマンドの組合せによって書き表すことができる．この場合，当該レイヤそのものを図示し，その上に “Min. Grid = n” といった注釈を添える．

エラーを報告するコマンドとして or コマンドが使われていた場合は，参照されている 2 つの派生レイヤを生成しているそれぞれのコマンドをエラー報告コマンドと見なしてこれまでに述べた中から該当するテンプレートを適用し，生成される 2 つの図を横に並べて表示する．

4. 評価・考察

4.1 実　　装

提案する手法に基づくプログラムを CPU：PentiumIII 1 GHz，主記憶容量：256 MB，OS：Windows 2000 Professional の計算機上で JAVA 言語を用いて実装した．プログラムはルールファイルパーサ部とメイン部に分かれており，前者は JavaCC を使い，2,000 行程度，後者も 2,000 行程度の記述となっている．

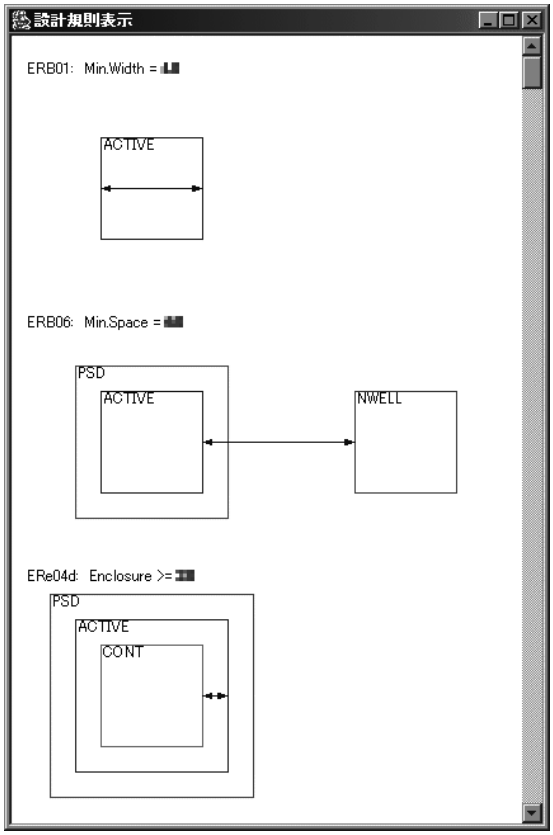


図 13 設計規則を表示したフレーム
Fig. 13 A frame displaying design rules.

4.2 表　　示

ユーザがこのツールにルールファイルを与えると，**図 13** のような設計規則を表示したフレームが開く（所要時間は数秒程度である）．設計規則の表示は，設計規則名，設計規則の説明，設計規則で規定する値，設計規則のグラフィック表示で構成されている．フレームには設計規則が並んで表示される．

モトローラ社 1.2 μm メタル 2 層 CMOS プロセスのルールファイルは全体で 250 行程度，対象となる設計規則部は 115 行，ベースレイヤ数 10，設計規則数（エラー報告コマンド数）は 70 個であった．これに対し開発したツールを適用したところ，65 個の設計規則については設計規則書とほぼ同様の表示が得られた．これらの内訳は，**表 3** のとおりである．

可視化に失敗した 5 個の設計規則はいずれも，我々の用意したテンプレートとは異なったコマンド列によって記述されていた．このため別のテンプレートが適用され不自然な図が表示されたが，これは設計規則書の図と見比べることで容易に判別できた．これら 5 個の設計規則それぞれについて人手で検討を行った結

表 3 モトローラ社 1.2 μm メタル 2 層プロセスのルールファイルで必要となったテンプレートの一覧表
Table 3 List of templates needed for rulefile for Motorola 1.2 μm double-metal process.

規定するルール	該当するルール数	可視化成功ルール数	テンプレート数
最小幅	8	8	1
最小間隔	26	26	1
最小重なり距離	16	16	1
最小突き出し	7	5	1
コンタクトサイズの規定	3	0	0
レイヤどうしの重なり禁止	1	1	1
最小グリッド	9	9	1
計	70	65	6

果，設計規則違反を正しく検出できるコマンド列であることが確認できた．そのうちの 1 個は設計規則自体にやや複雑な条件が含まれていたため提案手法による可視化は困難であると考えられるが，残りの 4 個は典型的な設計規則をやや変則的なコマンド列で記述しているものであった（ルールファイルの保守性の観点から，我々の用意したテンプレートどおりの記述に書き改めることがむしろ望ましいと考えられる）．以上のようにして，DRC ルールファイルの検証という目的を果たすことができた．

4.3 考　　察

4.3.1 抽出に対する考察

3.1 節にある派生レイヤ定義表は，派生レイヤの定義をベースレイヤと重なるか否かによって表現したものである．これは非常に単純な派生レイヤのモデリングではあるが，モトローラ社 1.2 μm プロセスのルールファイル中の設計規則を表示するには十分であった．他のプロセスのルールファイルに対しても，大半の設計規則で使われる派生レイヤはこの方法でモデリングできると考えられる．しかし，微細プロセスの中には Fat Wire Rule など，派生レイヤの定義の中にリサイジング演算や計量演算が使われるものも見られるため，派生レイヤのモデリング法を拡張する必要が生じると考えられる．

3.2 節の包含関係は，重なるレイヤ図形を表示する際の位置関係を決めるためのヒューリスティックであり，これによって，設計規則書により近い表示が得られる．しかし，特にヒューリスティック (2) と (3) から推定される包含関係はあらゆる場合に適用できるものとは限らない．モトローラ社 1.2 μm メタル 2 層 CMOS プロセスのルールファイルでは特に問題は生じなかったが，他のプロセスのルールファイルでも評価を行い，ヒューリスティックの改良や人手の介入などが必要ないかどうか検討する必要がある．

4.3.2 表示に対する考察

本ツールの表示は，設計規則を 1 つ 1 つ別々に配置したものとなる．しかし，一般に設計規則書は図 1 のように関連する設計規則が 1 つの図面に複合的に書かれている．一般的な設計規則書の体裁により近い表示を得るためには，表示アルゴリズムを大幅に変更する必要がある．しかし，本ツールの第 1 の目的はルールファイルを検証するために設計規則書と照合するのを支援することであるから，設計規則 1 つ 1 つが別々の図として表示されることは大きな問題とはならない．

4.3.3 テンプレートに対する考察

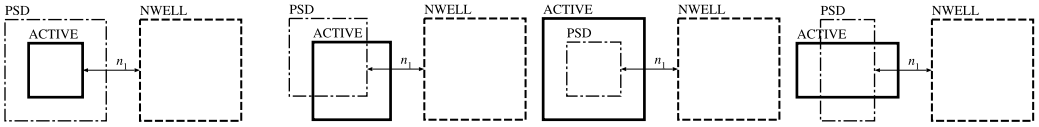
提案手法では，ルールファイルの中で設計規則を表すコマンド，たとえば最小幅を規定する width コマンド，最小間隔を規定する ext コマンド，最小重なり距離を規定する enc コマンドなどに対応する表示用のテンプレートを用意している．この手法は，これらのコマンドで使われる派生レイヤがどのようなものであるかにはあまり関係なく表示ができるという特徴がある．これは，1 つの設計規則を表す命令列全体に表示用のテンプレートを対応づけさせた従来の手法[3] よりも汎用性の点で優れている．これは設計規則数 70 のモトローラ社 1.2 μm プロセスのルールファイルに対して従来手法が 43 種のテンプレートを要したのに対し，提案手法は 6 種のテンプレートで 65 の設計規則の表示が可能になったことからもいえる．

4.3.4 表示の正しさについて

提案する手法では，ルールファイルからの情報抽出にかかる操作として，派生レイヤの定義の抽出，包含関係の抽出，設計規則テンプレートの選択の 3 つがある．このうち，派生レイヤの定義の抽出は and コマンドと not コマンドを検索するだけであるから誤りの入る余地はない．以下，残りの 2 つの正しさについて検討する．

4.3.4.1 包含関係の抽出の正しさ

包含関係の抽出はヒューリスティックである以上，誤認識が避けられない．しかし，包含関係情報の抽出は派生レイヤの定義をより実際のレイアウトに近い形で分かりやすく表示するためのものであり，誤認識がただちに致命的な表示誤りにつながるわけではない．たとえば図 3 の例で PSD と ACTIVE の包含関係が仮に誤認識された場合に得られる表示のバリエーションを**図 14** に示す．これらの図が PSD と ACTIVE の包含関係を表すために出力されたものでないことに注意されたい．これら 4 つの図はいずれも，PSD と ACTIVE が重なった領域から NWELL までの最小距離を規定しているという意味においては正しいとい



(a) 望ましい表示 (b) 望ましくない表示

図 14 包含関係認識ミスの影響
Fig. 14 Result of mis-recognition of enclosure relation.

える．

4.3.4.2 設計規則テンプレート選択の正しさ

開発したツールは，エラーを報告するコマンドがいずれのテンプレートにも正しく合致しない場合や，参照している派生レイヤの生成過程で想定外のコマンドが使われていた場合，図は表示されず，エラーメッセージを出力する．このため，設計規則を正しく表していないコマンド列が与えられた場合に，本来の設計規則を表すテンプレートに従ってあたかも間違いがないかのような表示をすることは原理的にありえない．ただし，図 11 (c) のテンプレートと同図 (d) のテンプレートの選択に限りヒューリスティックに認識された包含関係の情報が適用されるため，利用者は本来図 11 (c) の表示が得られるべきところで同図 (d) の表示が得られる可能性があることに留意しておく必要がある．

このほか，テンプレート自体の妥当性は入念に検討する必要がある．たとえば Cadence 社 Dracula の各コマンドには非常に多岐にわたるオプションがあり，オプションの組合せによってコマンドの意味内容が大きく変化する場合もあるのでテンプレート作成者はそのようなことをよく理解していることが必要である．

5. おわりに

5.1 まとめと今後の課題

本論文では設計規則検査で使用する DRC ルールファイルから設計規則を抽出してその可視化を行うアルゴリズムを提案した．この手法は，あらかじめルールファイルから派生レイヤの定義の抽出とレイヤどうしの包含関係のヒューリスティックによる推定をしておき，設計規則違反を報告するコマンドに基づいたテンプレートに派生レイヤの表示をあてはめることで設計規則の表示を行うものである．

本研究で開発したツールに，モトローラ社 1.2 μm メタル 2 層 CMOS プロセスのルールファイルを適用したところ，わずか 6 種類のテンプレートにより，70 個の設計規則のうち 65 個について設計規則書とほぼ同様の表示ができた．このためルールファイルの検証者は残りの 5 個の設計規則の記述の検討に注力できるようになった．

なお，今回開発したツールは Cadence 社 Dracula 用の DRC ルールファイルをターゲットにしているが，本論文で提案した手法の基本的アイディアは，Menter 社 Caribre 用の DRC ルールファイルなど，パターン演算の命令列で設計規則を表すタイプのものに広く適用可能であると考えられる．

今後の課題として，開発したツールをより微細なプロセスのルールファイルに適用し，提案手法の有用性を確認することや，ヒューリスティックの改良などを行うことがあげられる．

5.2 関連研究

STARC の原らは設計規則を記述するための新しい言語として SoDRML を提案している[6]．SoDRML は記述性に優れているうえ，専用のツールを用いて設計規則をグラフィック表示したり，主要な DRC ツール用のルールファイルに自動変換したりすることができる．このため，今後の新規プロセスは SoDRML で設計規則を記述することにより，個別の DRC ツール用のルールファイル作成や検証に必要な労力を大幅に軽減することができると考えられる．

本研究で開発したツールは，これまでデファクトスタンダードとして使われてきた Cadence 社 Dracula 用の DRC ルールファイルを直接解釈できることを最大の特徴としており，ルールファイルという資産の維持管理で有用であると考えられる．また，今回開発したツールに SoDRML 形式のルールファイルを出力する機能を付加することを現在検討中である．これにより，SoDRML の普及に寄与できるのではないかと考えられる．

謝辞 本研究を遂行するにあたり，東京大学大規模集積システム設計教育研究センターを通じ Cadence 社から提供された Dracula およびそのマニュアルと，同センターを通じ日本モトローラ（株）から提供されたテクノロジ情報を利用させていただきました．この場を借りて謝意を表します．

参考文献

1) Aho, A.V., Sethi, R. and Ullman, J.D.: *Compilers —Principles, Techniques, and Tools*, Ad-

dison Wesley, Reading, Massachusetts (1986).
2) Zima, H. and Chapman, B.: *Supercompilers for Parallel and Vector Computers*, Addison Wesley, Reading, Massachusetts (1990).
3) 北浦直樹，越智裕之，津田孝夫：DRC ルールファイルからの設計規則抽出とその可視化，信学技報，Vol.102, No.166, pp.1–6 (2002).
4) 長尾 真 ほか（編）：岩波情報科学辞典，岩波書店，東京 (1990).
5) 渡辺 誠, 池田邦博, 可児賢二, 大附辰夫：VLSI の設計 I，岩波書店，東京 (1985).
6) 原 浩幸，井上隆秀，中村忠彦：Design Rule Markup Language for STARC Open Design Rule Initiative, DA シンポジウム 2001, B10-1 (2001).

(平成 14 年 10 月 16 日受付)
(平成 15 年 3 月 4 日採録)

北浦 直樹

2002 年広島市立大学情報科学部情報工学科卒業．現在，同大学院情報科学研究科博士前期課程情報工学専攻在学中．DRC ルールファイルからの設計規則抽出とその可視化の研究に従事．

越智 裕之（正会員）

1989 年京都大学工学部情報工学科卒業．1991 年同大学院工学研究科修士課程情報工学専攻修了．1994 年同博士後期課程修了，博士（工学）．同年広島市立大学情報科学部情報工学科助教授，現在に至る．低消費電力設計，再構成アーキテクチャ，二分決定グラフ等の研究に従事．電子情報通信学会会員．

津田 孝夫（正会員）

1957 年京都大学工学部電気工学科卒業．1959 年同大学院工学研究科修士課程電気工学専攻修了．北海道大学工学部教授，京都大学工学部教授等を経て，1996 年広島市立大学情報科学部情報工学科教授，現在に至る．京都大学名誉教授．工学博士．自動ベクトル化/並列化コンパイラ，モンテカルロ法等の研究に従事．ACM，SIAM 各会員．