微細テクノロジ向け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）をDRCツールを用いて実行するためには，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り，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．この人手で作成されたルールファイルそのものの検証を支援するため，我々はルールファイルから設計規則を抽出して可視化することに取り組んできた．すでに我々は，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，そこで参照されるレイヤの表示アルゴリズムをそれぞれ用意し，少ないテンプレートで様々なルールファイル記述に対応することを狙った手法を提案し，その有用性を示した．本論文では上の手法をさらに改良し，より微細なプロセスへ対応するべく，表示するレイヤのモデリングに使用するパラメータの拡張と抽出データ，テンプレートを追加した手法を提案する．追加した抽出データは独立関係（2つのレイヤが重ならない），継承関係（一方のレイヤが他方のレイヤをもとに生成された），最小間隔，最小重なり距離である．提案手法に基づき可視化ツールを実装した結果，237個の設計規則を持つ0.13 μm メタル5層プロセスのルールファイルに対し，9種のテンプレートにより196個の設計規則が規則書とほぼ同様に表示された．

Extraction and Visualization of Design Rules from DRC Rule Files for Deep-submicron Technology

Naoki Kitaura,[†,☆] Hiroyuki Ochi[††] and Takao Tsuda[†]

In order to use a DRC (Design Rule Check) tool to perform physical layout verification, a tool-specific rule file which describes the design rule is required. A rule file is usually created manually, based on the design rule given by documents. To support the verification of the manually-created rule file itself, we have tried to extract and visualize the design rule from a DRC rule file. We have already proposed a method based on a display template corresponding to an error-reporting command and, independently, a display algorithm for reference layers, to cover various kind of rule file descriptions with small number of templates, and we have shown its effectiveness. In this paper, we will propose an improved method which is applicable for miniaturized technologies, based on enhanced parameters for medeling reference layers and additional extracted data and display templates. The additional extracted data are unoverlapping relation, extends relation, minimum space, and minimum enclosure. From experimental results using developed tool based on the proposed method, 196 out of 237 design rules of a 0.13-μm 5-layer-metal process are displayed similarly to the design rule manual using 9 display 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

† 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
†† 京都大学大学院情報学研究科通信情報システム専攻
Department of Communications and Computer Engineering, Graduate School of Informatics, Kyoto University
☆ 現在，株式会社メイテック
Presently with Meitec Corporation

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

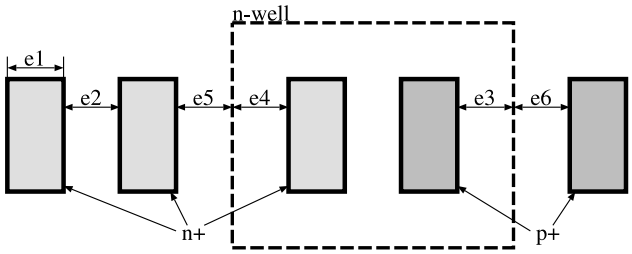


図 1 設計規則書の例
Fig. 1 An example of design rule manual.

が，設計規則を記述するための数学的に厳密な裏づ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．

ルールファイルと設計規則書の照合を支援するため，我々は 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研究してきた．まず，ルールファイルから設計規則を抽出するための素朴な手法として，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[1]，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．これは，同じ設計規則を表す命令列であってもその書き方は任意性があり，それらをカバーするために様々なテンプレートを用意する必要があるからである．そこで我々は，より少ないテンプレート数で多くの記述パターンに対応させるべく，(1) 設計規則そのものの表示アルゴリズムと，(2) 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ることを考えた．ここで (1) 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，(2) 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．この手法に基づき，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，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[2]．しかしこの手法では，微細プロセスにみられる太線規則や後光規則等に対応することが不可能であった．これらの規則に対応するためには，テンプレートを追加するとともに，派生レイヤの属性情報をよりきめ細かく抽出する必要がある．

そこで本論文では文献 2) の手法を改良し，より微細なプロセスへ対応するべく，表示するレイヤのモデリングに使用するパラメータの拡張と抽出データ，テンプレートを追加した手法を提案する．追加した抽出データは独立関係（2 つのレイヤが重ならない），継承関係（一方のレイヤが他方のレイヤをもとに生成された），最小間隔，最小重なり距離である．提案手法に基づき可視化ツールを実装した結果，STARC 社 0.13 μm メタル 5 層 CMOS プロセスのルールファイル（記述されていた規則 237 個）に対し，9 種のテンプレートにより 196 個（83%）が規則書とほぼ同様に表示された．

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[3]．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等を定めるものである．

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．

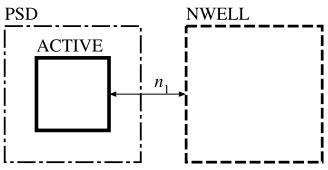
2.2 設計規則検査（DRC）とルールファイル

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」等であり，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等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[4] 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



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”.

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．

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，設計規則をルールファイルから抽出して可視化する素朴な手法を文献 1) で提案している．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，

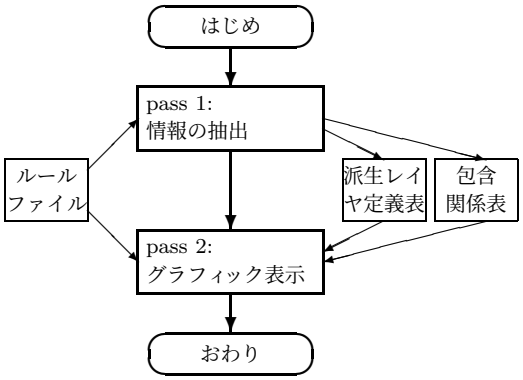


図 4 文献 2) の手法の流れ図
Fig. 4 Flowchart of the proposed method.

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，レイヤサイズ等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 1.2 μm メタル 2 層 CMOS 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．

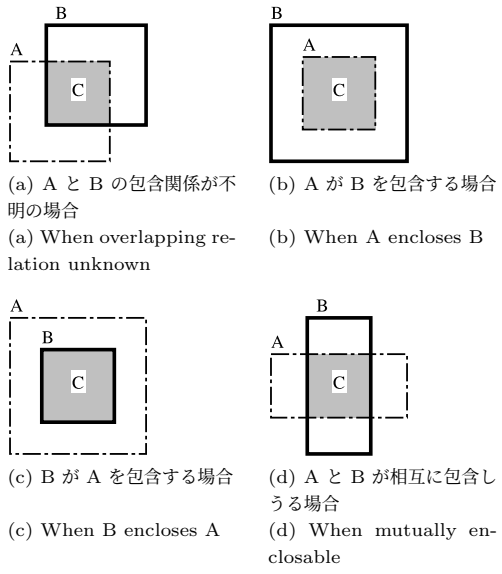
2.4 文献 2) の手法とその問題点

文献 1) の手法の問題点を解決するべく，文献 2) で我々が提案した手法の流れを**図 4** に示す．

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．ここで前者は，各派生レイヤがどのようなベースレイヤに対しどのように and コマンドや not コマンドを適用して生成されたものであるかという情報である．また後者はヒューリスティックを用いて推定されたベースレイヤどうしの包含関係☆である．包含関係を推定するためのヒューリスティックは以下の 3 つである．

(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



(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 が相互に包含しうる場合
(d) When mutually enclosable

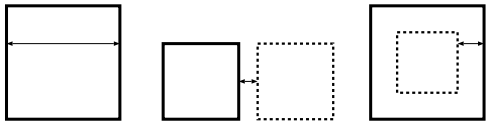
図 5 “and A B C” で得られたレイヤ C の表示
Fig. 5 Graphics for layer C derived from “and A B C”.

(3) not A B C
というコマンドがあれば B は A を包含する（ただし誤認識を避けるため，A，B がともにベースレイヤの場合に限り適用する）

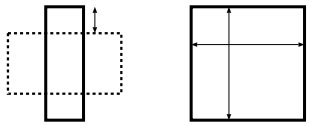
pass 2 では，エラーを報告するコマンドに対応するテンプレートを検索し，そのテンプレートに基づき設計規則の表示を行う．ここでエラーを報告するコマンドの引数となっているレイヤを適切に可視化するため，pass 1 で抽出された情報を援用する．文献 2) の手法が文献 1) と大きく異なるのは，pass 2 において，設計規則そのものの表示アルゴリズムと，その設計規則で参照されるレイヤの表示アルゴリズムを独立させた点である．これにより，少ないテンプレート数でより多くのルールファイル記述に対応することを狙っている．

派生レイヤの表示は，pass 1 で抽出された派生レイヤの定義情報およびレイヤどうしの包含関係情報を用いて行う．前者の情報により，ある派生レイヤ C が 2 つのベースレイヤ A，B の重なりとして定義されていることが明らかになった場合，この派生レイヤ C の表示は，**図 5** に示すとおり 4 種類のものが考えられる．この中から適切なものを選択するため，ヒューリスティックに推定された包含関係情報が用いられる．図 5 (d) は，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等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

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



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



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

図 6 設計規則表示用テンプレートの例
Fig. 6 Example of graphics templates for design rules.

用することにより表示を生成している．

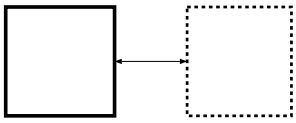
設計規則の表示は，エラーを報告するコマンドとしてしばしば用いられる各コマンドについて，あらかじめ用意されたテンプレートに基づき行う．テンプレートは，最小幅，最小間隔，最小重なり距離，最小突き出し長さ，コンタクトサイズ，重なり禁止，最小グリッドの 7 種類を用意した．代表的なものを**図 6** に示す．

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．

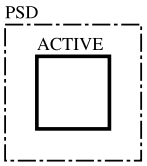
(1) （派生レイヤの定義）派生レイヤ PSUB はベースレイヤ PSD および ACTIVE の重なった領域である
(2) （レイヤどうしの包含関係）レイヤ ACTIVE はレイヤ PSD に包含される

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．この ext コマンドは最小間隔エラーを報告するものであり，これに対応する最小間隔規則の表示テンプレートは**図 7** (a) のとおりである（図 6 (b) を再掲）．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元に派生レイヤ PSUB を表すと図 7 (b) のようになる．一方，NWELL レイヤはベースレイヤなので，図 7 (c) のように表せばよい．これら 3 つより，図 7 (d) のような表示が生成される．

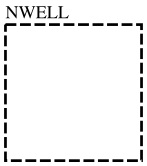
☆ Cadence 社 Dracula のルールファイルの場合，“output” が付されたコマンドがこれにあたる．



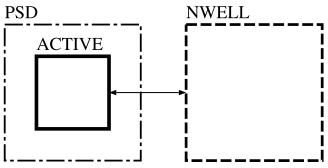
(a) 最小間隔規則の表示テンプレート
(a) Graphic template for Min. space



(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 7 図 3 から図 2 を生成する流れ
Fig. 7 Process for generating Fig. 2 from Fig. 3.

以上で述べた手法を 1.2 μm メタル 2 層 CMOS テクノロジのルールファイルに適用したところ，わずか 7 種類のテンプレートにより，70 個の規則のうち 65 個の規則を可視化することができた．しかし，この手法をより微細なプロセスに適用するためには以下の問題を解決する必要があった．

(1) 派生レイヤの定義を抽出する際，ベースレイヤの重なり関係しか考慮していない．つまり，and あるいは not コマンド以外を用いて生成された派生レイヤは表示することができない．たとえば太線規則では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ため，派生レイヤのモデリング法を拡張する必要がある．
(2) 包含関係を抽出するヒューリスティックとして，上に述べた 3 種類のものだけで十分であるとは限らない．
(3) 表示用テンプレートについても，最小面積規則や後光規則等，微細プロセスでみられるものを追加する必要がある．

3. 提案手法

本章で述べる提案手法の特徴を文献 2) の手法と比較して**表 1** に示す．提案手法の基本的な流れは，2.4 節の図 4 とほぼ同様であるが，提案手法は微細なプロ

表 1 提案手法の改良点
Table 1 Enhancements of proposed method.

	文献 2) の手法	提案手法
派生レイヤのモデリングと表示	and および not コマンドで生成された派生レイヤのみサポート．表示パターンは図 9 の (a)，(c)，(d)，(e) の組合せのみ．	select や size コマンド等で生成された派生レイヤもサポート．size コマンド等のパラメータも抽出．表示パターンは図 9 の (a)〜(f) の組合せ．
抽出される他の情報	包含関係（ヒューリスティック 3 種）	包含関係（ヒューリスティック 5 種），独立関係，継承関係，enc および ext コマンド
設計規則表示テンプレート	7 種（最小幅，最小間隔，最小重なり距離，最小突き出し長さ，コンタクトサイズ，レイヤどうしの重なり禁止，最小グリッド）	9 種（最小・最大幅，最小間隔，最小重なり距離，最小突き出し長さ，最小面積，レイヤどうしの重なり禁止，レイヤどうしの重なり必須，屈曲図形禁止，斜め図形の長さと最小幅）

セスの設計規則に対応するため，表示するレイヤのモデリングに使用するパラメータの拡張と抽出データ，テンプレートの追加がなされている．

3.1 抽　　出

3.1.1 派生レイヤの定義の抽出

and や not コマンド以外を用いて生成された派生レイヤにも対応できるよう，文献 2) から大幅に変更した．基本的には，派生レイヤの生成に用いられたコマンド（and，select 等）と引数の並びをそのまま記録し，後述の表示段階で解釈を行うように改めた．ただし，size コマンドと not コマンドについては以下に述べるとおり，抽出段階で情報を整理している．

3.1.1.1 size コマンド

あるレイヤの図形に対し，拡大，縮小を行う size コマンドが繰り返し使われている場合は，以下の情報を記録する．

(1) 拡大，縮小の対象となった元のレイヤ
(2) 一連の size コマンドによる累積的な拡大，縮小値
(3) 一連の size コマンドの過程での (2) の最小値

(2) と (3) は，太線規則関連で size コマンドが連続して使用される場合を想定したものである．ここで太線規則とは，一定以上の太さを持った図形に対し，同じレイヤの一般的な最小間隔規則よりも大きな最小間隔を要求する規則である．

以下，例をあげて説明する．**図 8** のコマンド列は，レイヤ M1 の図形に対しすべての輪郭を 1 ずつ縮小したものをレイヤ tmp1 とし，tmp1 を 1 ずつ拡大したものをレイヤ M1F2 とする記述である．この場合，tmp1 に関して保持される情報は (1) M1，(2) −1，(3) −1，M1F2 に関して保持される情報は (1) M1，(2) 0，(3) −1 となる．このコマンド列の 1 行目でレイヤ M1 の図形のうち幅が 2 以下のものは消滅するため，レイヤ M1F2 はレイヤ M1 の図形のうち，幅が 2 より大きいものとなるが，このことは M1F2 の (3) の値が負であり，その絶対値の 2 倍が 2 であることからただちに判定できる．

```
size    M1      by −1   tmp1
size    tmp1    by 1    M1F2
```

図 8 size コマンド使用例
Fig. 8 Example of size command.

3.1.1.2 not コマンド

あるレイヤの図形に対し，別のレイヤの図形と重なった領域を取り除く not コマンドが繰り返し使われている場合は，元の（取り除かれる側の）レイヤ 1 つに対し，取り除く側の複数のレイヤを一括して記録する．これにより情報がコンパクトになる．

3.1.2 包含関係の抽出

2.4 節で述べた 3 つのヒューリスティックに加え，以下の 2 つを追加する．

(3') not A B C output err
　というコマンドがあれば B は A を包含する
(4) and A B ab
　enc[T] ab B lt n output err
　というコマンド列があれば B は A を包含する

(3') は 2.4 節の (3) とほぼ同様であり，この記述がエラー報告する行であるかという点で異なる．(4) の 2 行目の 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 ab の輪郭までの距離を規定する記述である．この enc コマンド単体からは，2.4 節の (1) のヒューリスティックで包含関係が抽出される．(4) は enc コマンドの被包含側であるレイヤ ab が and コマンドによって生成され，さらに enc コマンドの包含側であるレイヤ B が and コマンドに使われているという記述である．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．

3.1.3 独立関係の抽出

レイヤ A の図形と，レイヤ B の図形が互いに重ならないとき，レイヤ A とレイヤ B は独立していると

呼ぶことにする．提案する手法では，レイヤどうしの独立関係をヒューリスティックに推定し，可視化の際にありえない状況が図示されることを防いでいる．このヒューリスティックとして下のものを導入する．

(1) and A B output err
というコマンドがあれば A と B は独立関係にある

この記述はレイヤ A の図形とレイヤ B の図形が重なった領域をエラーとして報告するものである．これより，レイヤ A の図形とレイヤ B の図形との重なりは許されない（独立関係がある）と推定される．

3.1.4 継承関係の抽出

レイヤ A の図形の中からある条件を満たすものをレイヤ A1 とするとき，レイヤ A1 はレイヤ A を継承すると呼ぶことにする．この継承関係は，後述する設計規則可視化の際に，エラーを報告する not コマンドの意味を区別する目的で使用される．

レイヤの継承関係は次の 4 つのコマンド列から抽出する．

(1) area A range n_1 n_2 A1
(2) area A eq n A1
(3) not A B A1
(4) width[L] A seleq n A1

(1) と (2) の area コマンドは，レイヤ A の図形のうち面積が条件を満たすものをレイヤ A1 とする記述である．(1) の条件は n_1 より大きく n_2 未満であるもの，(2) の条件は n であるものになる．(3) の not コマンドは，レイヤ A の図形のうち，レイヤ B の図形と重ならないものをレイヤ A1 とする記述である．(4) の width コマンドは，レイヤ A の図形のうち幅が n であるものをレイヤ A1 とする記述である．以上 4 つの場合に，レイヤ A1 はレイヤ A を継承しているとする．

3.1.5 enc コマンドおよび ext コマンドの抽出

ルールファイル中のすべての enc コマンドおよび ext コマンドを探し出し，これらを記録しておく．この情報の用途は後述する．

3.2 可 視 化

3.2.1 派生レイヤの表示

派生レイヤの表示は，3.1.1 項で述べた派生レイヤの定義情報に基づいて行われる．派生レイヤの定義に用いられたコマンド列に従い，**図 9** の描画コマンドを組み合わせて派生レイヤを表現する．そのアルゴリズムは以下のとおりである．

(1) ベースレイヤは，DRAW コマンドで表す．
(2) 後光図形を表す派生レイヤは，元のレイヤの DRAW コマンドを HALO_DRAW コマンドに置き換えて表す．
(3) (2) に該当しない and コマンドで生成された派生レイヤは，元のレイヤに ENCLOSE, CROSS，または AND_DRAW のいずれかのコマンドを用いて別のレイヤを重ね合わせて表す．
(4) ドーナツ図形を表す派生レイヤは，元のレイヤの DRAW コマンドを DOUGHNUT コマンドに置き換えて表す．
(5) or コマンド以外で生成された上記いずれにも該当しない派生レイヤは，元のレイヤに文字で説明を添えて表す．
(6) or コマンドで生成されたレイヤは，元の複数のレイヤを併記して表す．

(2) の後光（halo）図形とは，一定以上の太さを持ったあるレイヤの図形と，そこからその周囲の一定の範囲に伸びている同じレイヤの（必ずしも太くはない）図形を合わせたものである．最近の微細プロセスでは，メタルレイヤの太線規則が太線そのものだけでなく，そこから伸びた「後光」部分にも適用されることがあ

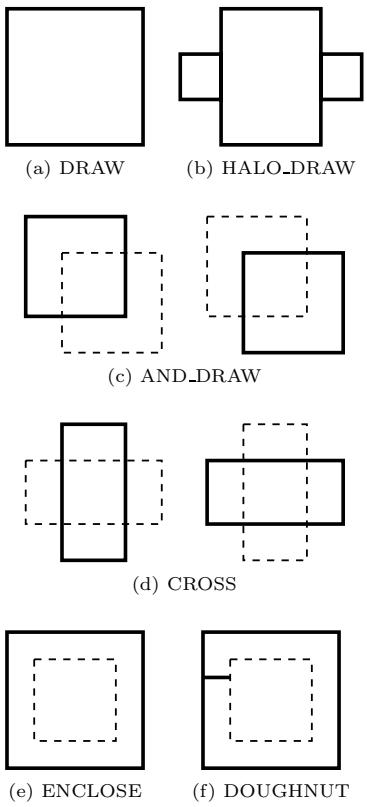


図 9 派生レイヤ表示用コマンド
Fig. 9 Commands for displaying a derived layer.

る．後光図形に対応する派生レイヤは，size コマンドで抽出された太線図形をさらに size コマンドで拡大し，これを元の同じレイヤと and するコマンド列によって生成されるので，このようなコマンド列が派生レイヤ定義情報に記録されていた場合は，HALO_DRAW コマンドが適用される．

(3) については文献 2) と同様，ヒューリスティックに抽出された包含関係情報を援用して ENCLOSE，CROSS，AND_DRAW の中から適切なコマンドを選択して表示する．

(4) のドーナツ図形とは，中がくり抜かれた図形である．最近の微細プロセスでは，一定の面積以下のドーナツ穴（たとえそれが最小間隔ルールを満たしていても）を許さないことがある．ドーナツ図形に対応する派生レイヤは，not コマンドを含む短いコマンド列で生成されるので，それに該当する場合は DOUGHNUT コマンドが適用される．

(5) に該当するものとして，たとえばあるレイヤ図形の中から面積が一定の範囲のものを抽出する area コマンドがある．この場合は，そのレイヤの図形に $S < 2$ 等と文字で説明を添えて表す．

3.2.2 設計規則の表示

設計規則の表示は，文献 2) と同様，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たものを使い行う．本論文では図 6 の (a)〜(d) に加え，微細プロセスに対応するため新たに 5 つのテンプレートを追加した．また，生成される表示を設計規則書により近いものにするための改良についても述べる．

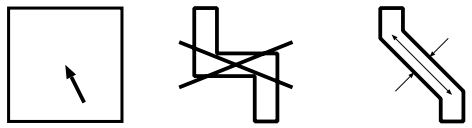
3.2.2.1 追加した設計規則表示用テンプレート

本論文で追加した設計規則表示用テンプレートの主なものを**図 10** に示す．

図 10 (a) は，ベースレイヤもしくは派生レイヤを 1 つ指定し，そのレイヤの最小面積を規定する最小面積コマンド（area）のテンプレートである．図中の矢印は，最小面積が規定されているレイヤを指す．

図 10 (b) は，ベースレイヤもしくは派生レイヤを 1 つ指定し，そのレイヤの図形が屈曲しているときにエラーを報告するコマンドのテンプレートである．この規則は微細プロセスにおいて屈曲ゲート禁止ルール等でしばしば見受けられるものである．

図 10 (c) は，一定以上の長さを持つ斜め配線について，ある幅以下であった場合にエラーを報告するコマンド列に対応づけられたテンプレートである．この規則は OPC を行っていると思われる微細プロセスで見受けられることがある．



(a) 最小面積　(b) 屈曲禁止　(c) 斜め図形の長さと最小幅

(a) Min. area　(b) prohibited bend pattern　(c) Min. width for diagonal pattern

図 10 追加した主な設計規則表示用テンプレート
Fig. 10 New graphics templates.

2 つのレイヤ図形の重なり部分を抽出する and コマンドがエラーを報告するようになっていた場合は，対象となる 2 つのレイヤを左右に並べて表示し，中央に「重なってはいけない」とテキストで説明を添える．

レイヤ A の図形からレイヤ B の図形と重なっていない部分を抽出する not コマンドがエラーを報告するようになっていた場合は，以下の 2 つに場合分けされる．

(1) （条件を否定する意味で使われた not）レイヤ A とレイヤ B が同一のベースレイヤを継承する派生レイヤであった場合は，レイヤ A の中でレイヤ B 生成中に課せられた条件を満たしていない部分がエラーであることを意味する．このため，レイヤ A を表示し，これに課せられるべき条件をテキストで表示する．

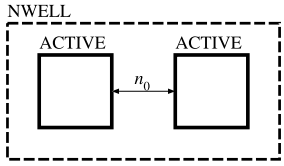
(2) （図形的な重なりを削除する意味で使われた not）レイヤ A とレイヤ B が異なるベースレイヤを継承する派生レイヤであった場合は，レイヤ A の図形がレイヤ B の図形上になければならないことを意味している．このため，2 つのレイヤを左右に書き並べ，レイヤ A がレイヤ B 上になければならない旨をテキストで表示する．

上の場合分けを行う際に 3.1.4 項で述べた継承関係の情報が参照される．

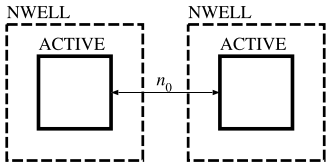
3.2.2.2 表示の改良

これまで述べたアルゴリズムでは，不自然な表示が得られることがあった．まずその例をあげる．

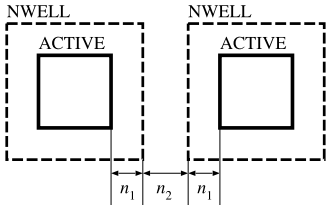
ウェル（NWELL）上の拡散層（ACTIVE）間の最小間隔 n_0 を規定する設計規則書の図面は**図 11** (a) のようになる．一方，これまで述べたアルゴリズムの場合，「NWELL 上の ACTIVE」という派生レイヤを左右にそれぞれ描き並べ，その間に ext コマンドのテンプレートに従って矢印を描画するため図 11 (b) のような表示になってしまう．しかし実際の設計で図 11 (b) のようなレイアウトを作成した場合，ACTIVE どうしの間隔は，ACTIVE に対する NWELL の最小重なり



(a) 設計規則書の図面
(a) Figure in a design rule manual



(b) 従来の可視化結果
(b) Graphics by conventional method



(c) (b) に関連する他の設計規則
(c) Related rules for (b)

図 11 従来の手法でうまく可視化されない例
Fig. 11 Bad example for conventional method.

距離 n_1 や，NWELL どうしの最小間隔 n_2 の制約を受けることになり，多くの場合，それらの和 $n_1+n_2+n_1$ は ACTIVE どうしの最小間隔 n_0 よりも支配的になってしまう．このため，n_0 が $n_1+n_2+n_1$ よりも小さい場合は，図 11 (b) のような表示ではなく，より設計規則書に近い図 11 (a) のような表示を生成することが強く望まれる．

そこで提案手法では，同一レイヤ間の最小間隔を規定する ext コマンドに対してテンプレートを適用する場合，そのまま図 11 (b) のように図示した場合に他の最小間隔規則等で与えられる制約（上の例の $n_1+n_2+n_1$ に相当する値）を求め，それが当該規則（上の例の n_0 に相当する値）よりも大きい場合は，そのまま表示せず図 11 (a) のような表示をするアルゴリズムを適用する．ここで他の最小間隔規則等で与えられる制約を検索するために，3.1.5 項で述べた情報が利用される．

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：Xeon 1.7 GHz × 2，主記憶容量：2 GB，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わかれており，前者は JavaCC を使い，2,000 行程度，後者は 5,000 行程度の記述となっている．

4.2 実 行 結 果

ユーザがこのツールにルールファイルを与えると**図 12** のようなフレームが開く（所要時間は数秒程度である）．フレーム内の上半分に並んでいる 2 つの子フレームは設計規則の一覧（左）と，そこで選択された設計規則を表示するもの（右）である（設計規則をマウスで選択してから表示されるまでの所要時間は 1 秒未満である）．さらに，右の表示は上下にわかれており上がグラフィック表示，下が説明（テキスト）となっている．図 12 の可視化結果は 3.2.2.2 で例示した設計規則を表しており，図 11 (a) のような望ましい表示が得られていることが確認できる．また図 12 は太線規則の表示例にもなっており，左側の拡散層を表す正方形中に「Width > 数値」という形式の添え書きが見える．また，**図 13** には後光規則の表示結果を示す．

STARC 社 0.13 μm メタル 5 層プロセスのルールファイルは全体で 760 行程度，対象となる設計規則部は 572 行，ベースレイヤ数 23，設計規則数（エラー報告コマンド数）は 237 個であった．このルールファイルは設計規則書に書かれている規則を誤りなく記述したものである☆．このルールファイルに対し開発したツールを適用したところ，196 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．表示されたものを含めこれらの内訳は，**表 2** のとおりである．

4.3 考　　察

4.3.1 可視化の正しさについて

提案手法による可視化は，派生レイヤの表示（3.2.1 項）と設計規則の表示（3.2.2 項）からなる．これらが正しく機能しなければ，誤りを含むルールファイルが設計規則書どおりに可視化されてしまう可能性もあり，ルールファイルの検証ツールとして用をなさない．以下，そのような可能性の有無について検討する．

4.3.1.1 派生レイヤの表示の正しさ

派生レイヤの表示は，派生レイヤの定義（3.1.1 項）および包含関係（3.1.2 項）の情報を用いて行われる．前者は当該派生レイヤを定義するコマンド列が本来持っている意味を解釈するだけなので，誤りの入る余地はない．一方後者は，ルールファイル全体を走査し

☆ 密度規則やアンテナ規則は今回開発したツールでは対象外である．実験で用いたルールファイルにはこれらの規則に関する記述は含まれていない．

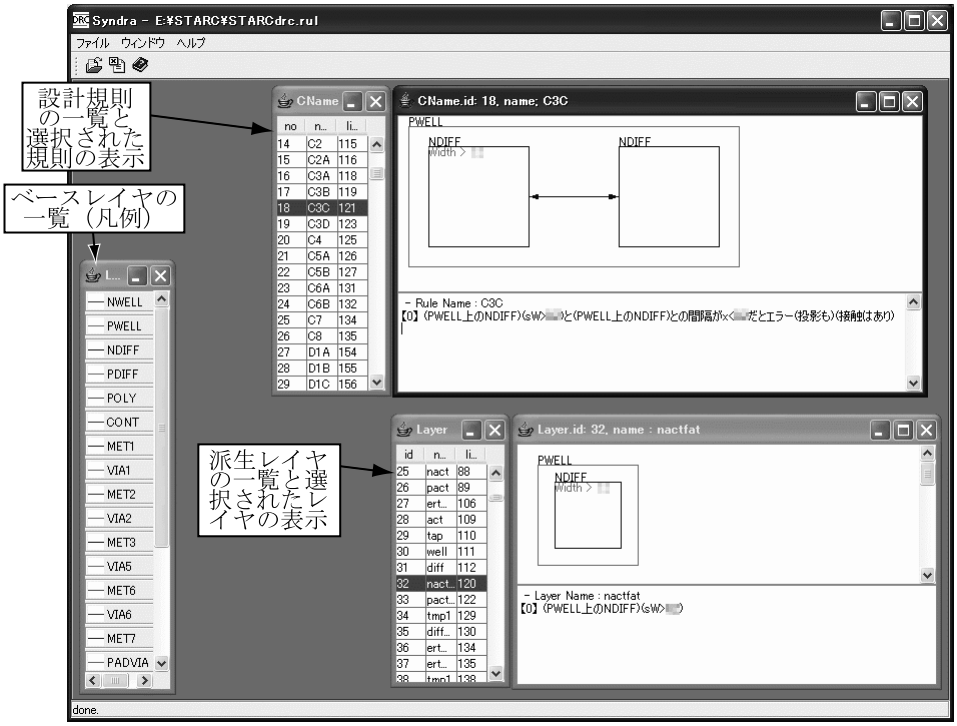


図 12 開発したツールの表示
Fig. 12 Graphics generated by our tool.

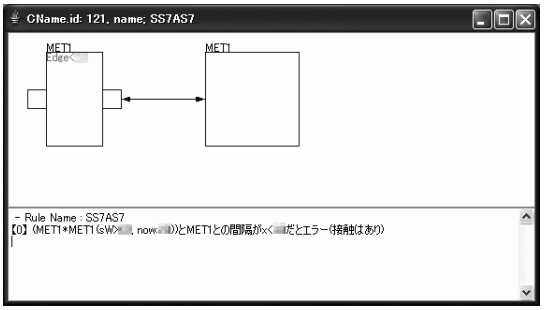


図 13 後光規則の表示
Fig. 13 Graphics for halo rule.

ヒューリスティックに推定するため，誤認識の可能性が排除できない．後者の情報を必要とするのは，and コマンドによって生成された派生レイヤであり，ヒューリスティックに推定された包含関係に基づき図 9 の (c)，(d)，(e) いずれかの描画コマンドが適用される．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える．

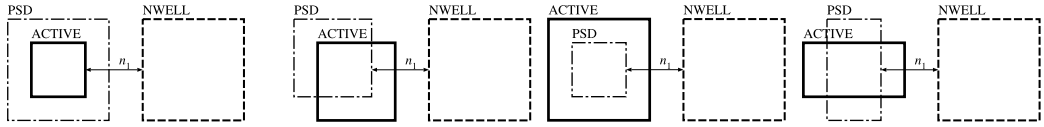
表 2 STARC 社 0.13 μm メタル 5 層プロセスのルールファイルで必要となったテンプレートの一覧表
Table 2 List of templates needed for rulefile for STARC 0.13 μm 5-layer-metal process.

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・最大幅	28	28	1
最小間隔	83	83	1
最小重なり距離	36	36	1
最小突き出し長さ	3	3	1
最小面積	13	13	1
レイヤどうしの重なり禁止	8	8	1
レイヤどうしの重なり必須	15	15	1
屈曲図形禁止	4	4	1
斜め図形の長さと最小幅	6	6	1
指定面積・指定幅	8	0	0
特殊な最小間隔	20	0	0
特殊な最小重なり距離	7	0	0
その他	6	0	0
計	237	196	9

4.3.1.2 設計規則の表示の正しさ

設計規則表示は，エラーを報告するコマンドに表示用テンプレートを対応づけて行われる．

まず，エラーを報告する 1 つのコマンドに対して複数のテンプレートが対応づけられていないものに関



(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.

しては，テンプレートを誤選択する余地がない．たとえば width コマンドは lt オプションをともなうとき，最小幅規則にのみ対応づけられる．同様に，最大幅規則，最小間隔規則，最小面積規則，レイヤどうしの重なり禁止規則，屈曲図形禁止規則，斜め図形の長さと最小幅規則もエラーを報告するコマンドおよびそのオプションだけでテンプレートが確定する．

次に，not コマンドは 3.2.2.1 で述べたように，あるレイヤの図形中で特定の条件を満たしていない部分をエラーとして検出するために使われる場合と，異なるレイヤが図形的に重なっていない部分をエラーとして検出する場合がある．not コマンドがどちらの意味で使われているかに関しては，not コマンドの引数となっている 2 つのレイヤが同一のベースレイヤに由来していれば前者，異なるベースレイヤに由来していれば後者の意味になる．この判定は，継承関係を用いて行われる．今回の実装では継承関係の抽出は 3.1.4 項で述べたとおり 4 つのヒューリスティックのみで行っており，継承関係の見落としがありうる．つまり，前者のテンプレートが適用されるべきところに後者のテンプレートが適用される可能性がある．ただし，この誤認識は同一レイヤがあたかも異なるレイヤであるかのように表示されるため，設計規則書と明らかに異なる不自然な表示となり，すぐにそれと分かる．

enc コマンドに関しては，文献 2) と同様に，ヒューリスティックに推定された包含関係に基づいて図 6 (c) の最小重なり距離のテンプレートと図 6 (d) の最小突き出し長さのテンプレートのいずれかが適用される．このため包含関係の誤認識が生じれば，これら 2 つのテンプレートの選択を誤ることになる．この点については利用者が留意しなければならない．表 2 より，これに該当する設計規則は 39 個（全体の 16.5%）である．この誤認識は提案したヒューリスティックが想定しなかったテクノロジのルールファイルのほか，誤認識を誘発する記述誤りを含むルールファイルや，認識に必要な記述が含まれていない未完成のルールファイルで生じる可能性がある．

4.3.2 可視化に失敗した設計規則について

表 2 より，可視化に失敗した設計規則は全体のわずか 17% であった．可視化に失敗した設計規則はいずれも多数のコマンドを組み合わせたり，特殊なコマンドオプションを利用しなければ検出することができない複雑な設計規則であった．たとえばトランジスタの end cap 周辺に限り field poly と拡散層との最小距離を厳しく規定している場合や，一定の条件下でコンタクト周囲の導体層の最小重なり距離を緩和している場合等である☆．これらは記述の任意性が高く，テンプレート化はなじみにくいと考えられる．

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 ルールファイルと設計規則書の照合を支援するべく，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について述べた．本論文で提案した手法は，我々が文献 2) で提案した手法にヒューリスティックの追加，テンプレートの追加，さらには太線規則等の微細プロセスにみられる設計規則への対応に必要となる各種情報の抽出を盛り込んでいる．

本研究で開発したツールを，STARC 社 0.13 μm メタル 5 層 CMOS プロセスのルールファイルに適用したところ，わずか 9 種類のテンプレートにより，237 個の設計規則のうち 196 個（83%）について設計規則書とほぼ同様の表示が得られた．ヒューリスティックを援用して表示される最小重なり距離規則と最小突き出し長さ規則について注意を払う必要は残されているが，ルールファイルの検証者は主として可視化に失敗した残り 17%の設計規則記述の検討に注力できるようになったと我々は考えている．

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 ルールファイル等，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．

また，今回開発したツールはルールファイル中の個々

☆ 特定のプロセスに関する詳細な情報の開示は容赦願いたい．

の規則を設計規則書と照合することだけを目的としており，表示は素朴で必ずしも見やすいとはいえないが，より実際の設計規則書に近い表示が得られるようになれば，設計規則書の図面を自動生成するためのツールとして役立つ可能性もある．設計規則抽出アルゴリズムに関しても，各規則について推奨される標準記述だけを本ツールが自動的に認識できるようにチェックを厳しくすれば，ルールファイルを作成する際に可読性や保守性の向上に役立てることができると考えられる．

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[5]．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）および（株）半導体理工学研究センター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) 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).
2) 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，情報処理学会論文誌，Vol.44, No.5, pp.1255–1265 (2003).
3) 長尾 真ほか（編）：岩波情報科学辞典，岩波書店，東京 (1990).
4) 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).
5) 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 16 年 11 月 15 日受付)
(平成 17 年 4 月 1 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．2004 年同大学院情報科学研究科博士前期課程情報工学専攻修了．同年株式会社メイテック入社，現在に至る．

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．DRC ルールファイルからの設計規則抽出とその可視化は，広島市立大学在職中に取り組んだテーマの 1 つである．2004 年京都大学大学院情報学研究科通信情報システム専攻助教授，現在に至る．再構成アーキテクチャ，低消費電力設計，算術演算回路等の研究に従事．電子情報通信学会，IEEE 各会員．

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 年より 2003 年まで広島市立大学情報科学部情報工学科教授．京都大学および広島市立大学名誉教授．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．