RC-009

DRC 検証作業の負荷を軽減するシステムの開発
Development of a System to Reduce the Load of DRC Verification

亀井 智紀†‡　　渡邊 孝博‡
Tomoki　Kamei　　Takahiro　Watanabe

1. はじめに

LSI のレイアウト検証の一つである DRC（Design Rule Check）検証は，デザインマニュアルに記載されたルール通りにレイアウトデータが作成されているかをチェックするものである[1]．しかし，レイアウトデータの大規模化に伴い DRC 検証を人手で行うことは実質不可能で，DRC ツール[2]と呼ばれる EDA（Electronic Design Automation）ツールを用いて検証を行うことが一般的である．DRC ツールがルール違反になっている箇所（以下，エラー図形と呼ぶ）を検出すると，その部分をレイアウト設計データの標準形式である GDSII（Graphic Database System）フォーマット[3]のデータやテキストデータとして出力する．作業者はエラー図形が検出されれば設計工程まで戻りレイアウトデータを修正する．しかし，DRC 検証を行う作業者と設計を行う作業者は異なることが多く，その場合は DRC 検証の作業者からエラーのレポートが設計者に届けられる．その際にレイアウトデータ上のエラー箇所の画像があれば，設計者はエラー原因を把握しやすい．そこで DRC 検証作業者は，ビューワと呼ばれる EDA ツールを用いて，DRC ツールから出力されたエラー図形をレイアウトデータに重ね合わせてエラー箇所の画像を作成し，保存する．ただ，エラー図形は大量になる場合もあり，その操作自体が DRC 検証の作業者にとっては負担になっていた．そこで筆者らはバッチ処理で自動的にレイアウトデータにエラー図形を重ね合わせ，画像保存を行うシステムを作成した．

ところで，エラー図形はレイアウト設計単位の領域であるセル内の特定箇所に集中することもあれば，セル全域に分布することもある．作業者が目視でエラー図形の確認をする場合，先ずセル全体が収まる画面（以下，FIT 画面と呼ぶ）で俯瞰的に眺め，エラー図形の個数や分布を確認する．しかし，FIT 画面では，個別のエラー箇所は小さすぎてエラーの内容を目視で確認することが困難であるため，作業者がビューワの画面を見ながら該当箇所を拡大し確認していくのが通常の手法である．一方でバッチ的にエラー箇所を画像保存する場合，拡大して保存する画像の領域をいかにして決定するかという問題がある．最も一般的な手段は一つずつのエラー図形を拡大して保存していく方法である（図 1 右上）．ただ，一カ所ずつ個別に画像を作成するとエラー図形が多いときは，画像の数も膨大になり一つずつ確認することはかえって煩雑になる．また冗長な画像ファイルが大量に生成されることでハードディスクなどの記憶装置の資源を圧迫することにもつながる．とはいえ，従来のバッチ処理では人間が判断するような“適度なまとまり”という判断は難しく，FIT 画面で画像を取得するか

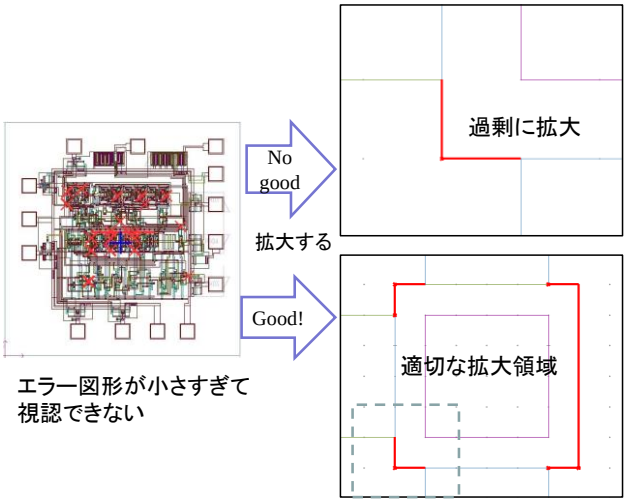

図 1. FIT 画面でエラー図形を重ねている表示（左），一つの図形を拡大した表示（右上），適切なグルーピングを行った表示（右下）

Figure 1 Display (left) of error figure overlaid on the FIT display, one figure displayed in enlarged state (top right), display after appropriate grouping (bottom right).

あるいはエラー図形を 1 か所ずつ個別に拡大した画像を取得するか，どちらかの方法にならざるを得なかった．

今回，筆者らが開発したシステムでは階層的クラスタリング法[4]を用いて，エラー図形を適度な領域でグループ化し画像保存する点が特長である．また，エラー図形が辺(edge)の場合とポリゴンの場合の両方に対応できるデータ構造を工夫した．これらの提案により，エラー図形一カ所ごとに画像保存した場合と比較して，大幅に取得画像を少なくすることができた．また，DRC ツールの出力はセルごとに行われ，TOP セル以外の下位階層のセルはその下位セルでの LOCAL 座標系でエラー図形が出力される場合がある．しかし，作業者としては IP(Intellectual Property)セルと別階層の配線の接続状況を確認したいこともあり，個々のセル内のエラーも TOP セルから見た画像で見たいという要求があった．このような要求に対応するため，座標変換せずに，下位階層のセル単位で図形を TOP セルに重ね合わせることにした．これによりセル内の図形を逐一 TOP 座標系に変換する手法に比べて画像表示にかかる計算量を大幅に軽減できた．図 2 はセル B が，その親セルであるセル A，さらにその親セルの TOP からどのように見えるか俯瞰した例である．以上のような工夫により，提案システムは DRC および LSI 設計に関わる作業者の負担を大幅に軽減し，効率的な設計検証が可能になった．

† TOOL 株式会社,TOOL Corporation
‡ 早稲田大学大学院情報生産システム研究科,The Graduate School of Information, Production and Systems, Waseda University

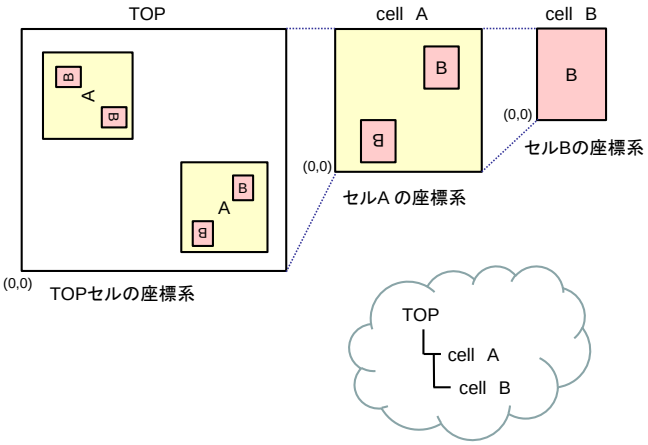

図 2.セルの座標系

Figure 2 Coordinate system of cell.

以下，2 章で本システムに概要について述べ，3 章では本システム処理内容の詳細について述べる．4 章では本システムの特長である階層的クラスタリング法について述べ，5 章で結果を評価し，6 章でまとめる．

2. システム構成と処理フロー

2.1 システム構成

2.1.1 計算機環境

本システム開発に使用した計算機環境を表 1 に示す．

表 1.システム構成

Table 1 System environment.

CPU	48 x AMD Opteron 6344 (2.54GHz)
Memory[GB]	512
OS	Linux3.10.0-123.9.3.el7.x86_64

2.1.2 使用ツール

DRC ツールは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し Mentor Graphics 社 Calibre [5]を使用した．本システム開発には TOOL 社 LAVIS-plus[6]を使用した．Calibre から出力される ASCII 形式のエラーデータベースファイルを GDSII に変換するツールとしては同社 lverrdb2gds[7] を使用している．本システムを開発するために LAVIS-plus に備わっている API(Application Program Interface)を介して Python 言語[8]にてプログラミングを行った．

2.2 システム概要

本システムの大まかな処理フローを図 3 に示す．本システムはバッチ処理を想定しているので，システムに必要な情報を与えるファイルを事前に用意する．この設定ファイル（Configuration file）をシステム起動後に読み込ませると自動的にエラー図形とその周辺を画像ファイル形式（PNG, JPEG, BMP 等）にして出力する．

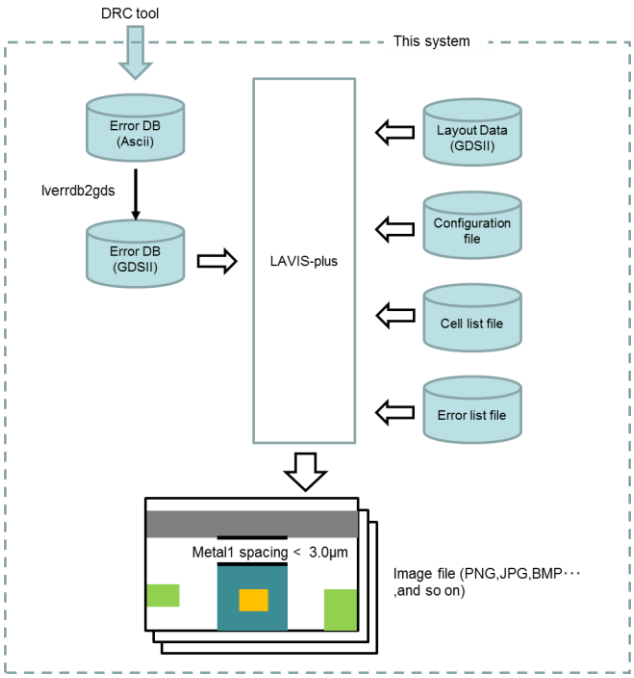

図 3.システム概略図

Figure 3 A System general flow.

(1)TARGET-FILE	/home/xxx/data.gds
(2)ERROR-DB-FILE	/home/xxx/drc.db
(3)CELL-LIST-FILE	/home/xxx/cellist.txt
(4)ERROR-LIST-FILE	/home/xxx/errorlist.txt
(5)WINDOW-MARGIN	2
(6)ERRORFLG-COLOR	Green
(7)ERRORFLG-WIDTH	3
(8)OUTPUTFILE-FORMAT	PNG
(9)GROUP-SIZE	10
(10)IMAGE-SIZE	500x500
(11)OUTPUTDIR	/home/xxx/output

図 4.設定ファイル書式

Figure 4 Format of the configuration file.

設定ファイルとして図 4 の形式を定義する．以下，この設定項目にしたがって入力ファイルの説明をする．

(1)TARGET-FILE はレイアウトデータの GDSII ファイルを指定する．本システムが起動する以前に DRC は終了している必要がある．

(2)ERROR-DB-FILE は DRC の実行結果である．DRC を実行すると ASCII 形式のエラーデータベースファイルが作成され，その中にエラー図形の頂点座標情報が記載されている（図 5）のでそのファイル名を指定する．最終的にこの ASCII 形式のファイルは GDSII 形式に変換される．詳細は 3 章に記す．

(3)CELL-LIST-FILE は図 6 のようにセル名を 1 列または 2 列で羅列して記載し，1 列目に書かれたセル内のエラー

```
TEST6 1000
R1.5
4 4 3 Nov  2 16:42:07 2014
Rule File Pathname: drc.rul
Nwell enclosure of Ndiff < 0.5um
(Substrate/Well contact Active to Nwell edge)
e 1 2
CN TEST6 c
641000 328000 641500 328000
641000 328000 641000 328500
                :
R6.1
1 1 2 Mar  2 16:42:07 2014
Rule File Pathname: drc.rul
Poly Width for defining Gate length < 4.00um
e 1 2
CN XDFF1 c
44000 46000 44000 98000
46000 46000 46000 101464
```

図 5.ASCII エラーDatabase 書式

Figure 5 Format of the ASCII Error-DB file.

図形が画像保存の対象となる．1 列目に単にセル名のみを記述した場合は，そのセルを開いた状態で FIT 画像とセル内部のエラー画像を保存する．2 列目に記述できるセル名は TOP セル名である．TOP セルから見た FIT 画像とセル内部のエラー画像を保存する． 詳しくは 3.1 に記す．

(4)ERROR-LIST-FILE は図 7 のように，画像に出力するエラールール名とその表示色を定義する．

(5)WINDOW-MARGIN は画像保存する際，セル枠からどの程度領域を広げて保存するかをミクロン単位で指定する．

(6)ERRORFLG-COLOR はエラー図形を描画する際，エラーリスト内で表示色が定義されていなかったときのデフォルト色を指定する．

(7)ERRORFLG-WIDTH はエラー図形を描画する際の線幅を 1 から 5 の 5 段階で指定する．

(8)OUTPUTFILE-FORMAT は出力画像のフォーマット種別（PNG，JPEG，BMP など）を指定する．

(9)GROUP-SIZE は作業者が指定するクラスタ間の結合距離を指定するが，詳しくは 4.2 に記す．

(10)IMAGE-SIZE は出力画像の大きさをピクセル単位で指定する．

(11) OUTPUTDIR はエラー画像の出力先を指定する．

以上の設定ファイルの条件下で，レイアウトデータと DRC エラー図形が読み込まれ，FIT 画面での表示が可能になる．

3. システム内部処理

3.1 起動からエラー図形セルの重ね合わせまで

本システムは Linux のシェル上から設定ファイル名を指定して LAVIS-plus を起動させる．先ず設定ファイル内の各項目値を解釈した後，LAVIS-plus 付属のユーティリティ lverrdb2gds を使用して，DRC ツールから出力された ASCII 形式のエラーDB ファイルを GDSII 形式に変換する．

```
TOP
XDFF            TOP
XTFF
BUFF10          TOP
PAD2
```

図 6.セルリストファイル書式

Figure 6 Format of the cell list file.

```
R1.5    RED
R2.1
R2.6    BLUE
R4.1
R6.1    RED
R6.2    YELLOW
R6.4    CYAN
R6.6    WHITE
R9.8    YELLOW
```

図 7.エラーリストファイル書式

Figure 7 Format of the error list file.

図 8.DRC から出力されたエラー図形（GDSII）

Figure 8 Output error figures from DRC.

この GDSII ファイルは図 8 のようにエラー図形のみの情報で構成されており，これと GDSII 形式のレイアウトデータとを重ね合わせてエラー箇所の確認作業ができる．その際に図 5 内のエラー内容の文字列（5−6 行目）も GDSII の TEXT へ変換しておく．それによりレイアウトデータに重ね合わせた際にエラーの原因が容易に判別できる（図 9）．次にセルリストファイルを読み込み，セルの記述順に以降の処理をするが，その際に LOCAL セル指定（1 列目のみ記載）か TOP セル指定（2 列目も記載）かにより，処理が分かれる．

3.1.1 LOCAL セル指定の場合

レイアウトデータからセルリストファイル 1 列目に記載されているセルを開く．次に GDSII 形式に変換されたエラーDB ファイルから該当セルを見つけて重ね合わせる．

3.1.2 TOP セル指定の場合

レイアウトデータからセルリスト 2 列目に記載されているセルを開く．このセルは必ず TOP セルである．TOP セルを開いたら，TOP セルに引用されている下位セルの配置点に関する情報を取得する．配置点に関する情報とは OFFSET（位置情報）（X,Y），ANGLE（角度），MIRROR（反転有無），SCALE（倍率）である．これらの情報をすべて加味して，TOP セルに 1 列目に記載されているセル名を重ね合わせる．

3.2 FIT 画像とクラスタリング後の画像保存

3.1 までレイアウトデータにエラー図形が重ねられた画面が作成されている． 3.1 において LOCAL セル指定により重ね合わせ画像を作成した場合は図 9 のようなイメージになっている．一方，TOP セル指定により重ね合わせ画像を作成した場合は図 10 のように対象セルだけでなく，その周辺部分も画像内に含まれる．周辺部分をどの程度画像保存の対象にするかは設定ファイルの WINDOW-MARGIN の項目で設定する．保存領域が決まれば PNG や JPEG などの指定した画像フォーマットで保存する．ところで，エラー図形はセル内のある箇所に塊となって出現する場合があり，そのような場合にエラー図形を一つずつ拡大して画像保存すると必要数以上の画像が作成されてしまい，後から確認する上で効率が悪い．例えば，図 9 や図 10 はエラーの図形数としては 2 であるが，それぞれ拡大して 2 枚の画像を作成しても，もう一つの図形が必ず含まれ，どちらもほぼ同じ画像になる．このようなときは次章で述べるクラスタリングを用いて，2 つを包含するような領域で 1 枚の画像を保存する．

4. 階層的クラスタリング法

膨大な数のエラー図形を適度な数にグループ分けして個々に保存するためにクラスタリングを行う．このとき，適切なグループ数は，エラー図形の分布度合に依存するので，前もって決めることは困難である．そこで今回，筆者らは階層的クラスタリング法を採用してエラー図形のクラスタリングを行った．以下では階層的クラスタリング法と本システムに実装したアルゴリズムについて述べる．

4.1 基本アルゴリズム

階層的クラスタリング法は個体を一組ずつ結合して小さなクラスタから次第に大きなクラスタにしていく手法である．

通常，2 次元ユークリッド空間でクラスタリングを行う場合，個体は座標点(x,y)になるが，今回は DRC ツールが出力するエラー図形である edge または polygon の二種類が対象となる（図 11）．edge は両端点の座標，polygon は各頂点の座標で表現されているので，クラスタリングを行うための代表点として，edge に関しては 2 点の中点の座標を，また polygon では外接矩形の中心点の座標を使用する．

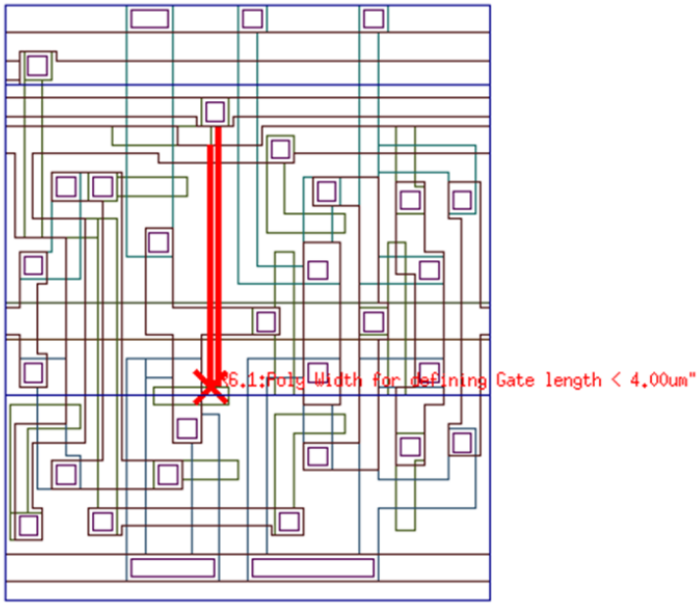

図 9.エラー図形とレイアウトデータ上の特定セルとの重ね合わせ（LOCAL セル指定）

Figure 9 Overlay of error figures and layout data (specified LOCAL cell).

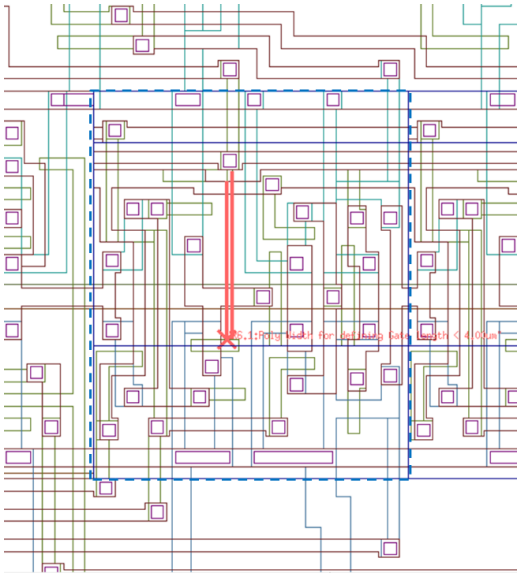

図 10.エラー図形とレイアウトデータの重ね合わせ（TOP セル指定，点線部が一つのセル）

Figure 10 Overlay of error figures and layout data (specified TOP cell).

階層的クラスタリング法のアルゴリズムは次のような手順からなる[9].

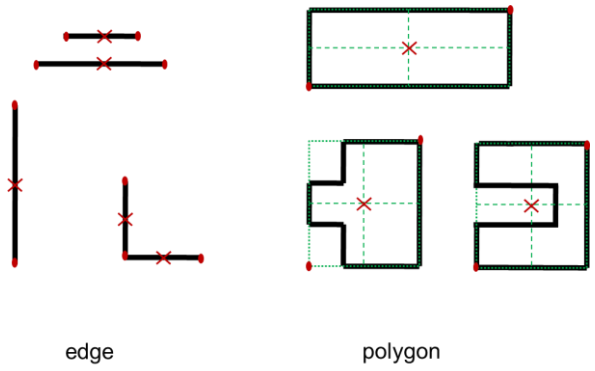

図 11.DRC ツールから出力されるエラー図形種別

Figure 11 Error figure types that are output from DRC tool.

[手順１] 初期状態として，n 個の個体それぞれが，一つのクラスタを形成しているとする．したがってクラスタの個数 K は K=n とする．

[手順２] K 個のクラスタの中で最も距離の小さい対を求め，それらを一つに併合して新しいクラスタとし，K=K-1 に更新する．

[手順３] K>1 ならば手順２へ戻る．そうでなければ終了する，

図 12 および図 13 にエラー図形を処理する場合のクラスタ Ci と Cj の併合前後の距離の様子を示す．図 11 で保持していた 2 頂点から中点×を求めてアルゴリズムの手順２を行い，クラスタを作成する．その際に凹図形では中点が polygon 内に含まれないこともあるが，システムが表示領域を決定するという点では問題はない．図 12 では Ci, Cj, Ck 3 つのクラスタが作成されている．クラスタは個々の要素の外接矩形であり，その対角頂点の座標のみを保持している．矩形の中点座標間のユークリッド距離 Ci-Cj 間，Ci-Ck 間，Cj-Ck 間で距離を測定し，最も距離の短い Ci と Cj のクラスタを結合する．図 13 で新しいクラスタ Ci∪Cj が作成され，外接矩形の対角頂点座標を更新する．この場合に必要なのはクラスタ Ci とクラスタ Cj の外接矩形の座標のみであり，各クラスタ内の要素の情報（座標）は不要である．次に新しく作成されたクラスタ Ci∪Cj とクラスタ Ck との距離を同様に測定し，最終的にクラスタ数が 1 になるまで繰り返す．

4.2 最終クラスタ数とデンドログラム

クラスタ生成のプロセスはシステム内部で作成する樹形図（デンドログラム）と呼ばれる二分木（tree）によって表される．デンドログラムは各個体を末端の葉（leaf）とし，クラスタが生成されるごとに対応する枝を結合していくので，クラスタ形成の全過程が示されている．また枝が結合される座標は，クラスタの結合レベルに対応している [4]．例えば図 12，図 13 のクラスタリングをデンドログラムにすると図 14 のようになる．ここで結合距離（merge distance）がクラスタを結合したときのユークリッド距離になる．クラスタ数が最終的に 1 になるまで実行した場合は，この結合距離の最も長い枝でカットする．

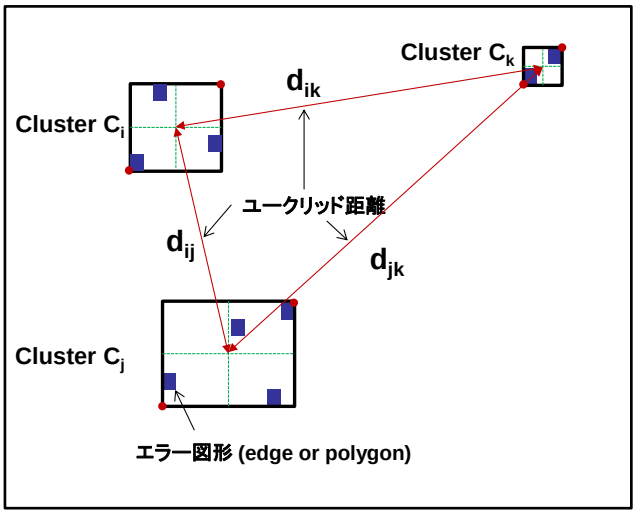

図 12.階層クラスタリング法　実行前

Figure 12 Before merging the clusters.

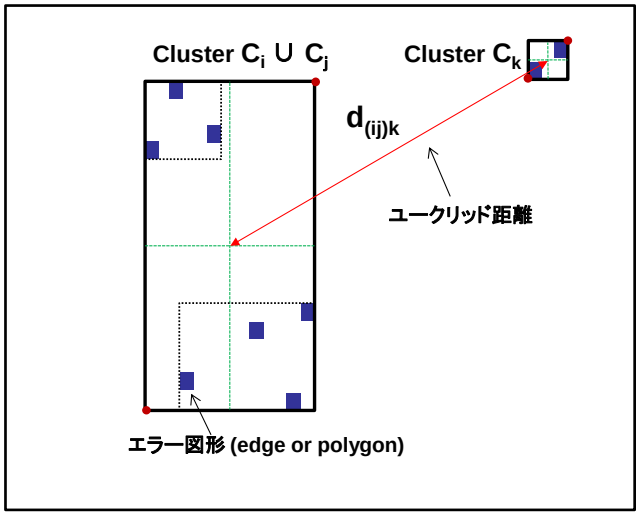

図 13.階層クラスタリング法　実行後

Figure 13 After merging the clusters.

デンドログラムをカットするとは，そのレベルに至るまでにできたクラスタを出力することになる．あるいは，結合距離を作業者が事前に指定できるとすれば，アルゴリズムを途中で中断しその時点でのクラスタを出力することができる．その場合はクラスタが 1 になるまで実行しなくてよいので，先に述べた処理よりも高速になる．そこで，設定ファイル上の GROUP-SIZE で作業者が結合距離を指定した場合は，その結合距離でクラスタを行った結果を最終結果とする．

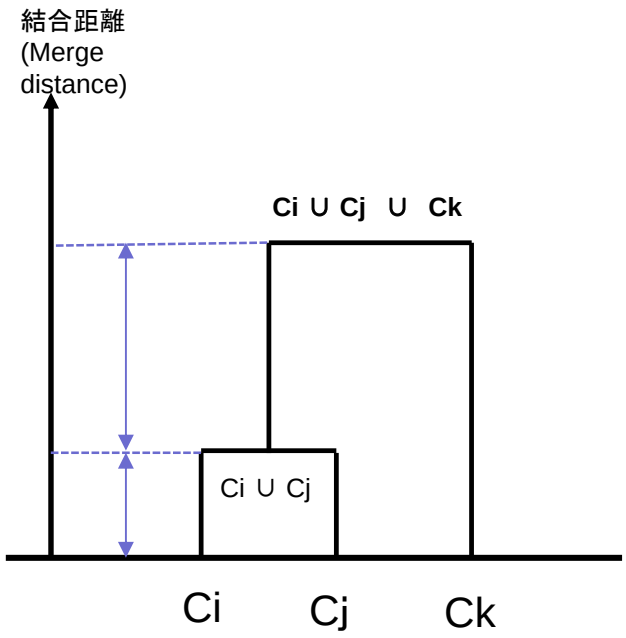

図 14. デンドログラムによるクラスタリング結果の表示

Figure 14 Display the results of the clustering by dendrogram.

5. クラスタリング法の評価

本システムを実行して作成された画像の一つを図 1 右下に示した．この画像内には edge で 7 個のエラー図形が存在しているが，クラスタリングを行うことで 7 個の図形を 1 枚の画像に集約できることが確認された．従来はエラー図形ごとに画像を作成していたため，edge の図形単位で 7 枚の画像が出力されていたため，出力画像数は大幅に削減された．

ここで，クラスタリングを実行した場合の本システムの処理時間について述べる．結果を表 2 に示す．クラスタリングのみの処理時間に着目すると GROUP-SIZE を設定した場合の方が高速になっている．一方で処理全体の時間は逆転しているが，これは画像の出力枚数が多くなるに伴い，保存処理の時間が増加することによる．

表 2 エラー図形数と処理時間の関係

Table2 The relationship between the processing time the number of error figures.

エラー図形数	58		306	
GROUP-SIZE(um)	10	無指定	10	無指定
出力画像数	16	4	147	6
CPU 時間(s)（クラスタリング）	0.136	0.191	0.870	1.850
CPU 時間(s)（処理全体）	1.152	0.498	9.676	3.137

6. むすび

本論文では，DRC ツールの出力結果であるエラー図形をレイアウトデータに重ね合わせて画像を保存するシステムの提案を行った．一連の作業はすべてバッチ制御で自動実行されるため，作業者が個々に操作をして画像を保存する場合と比較して，大幅に作業が軽減される．本システムでは下位階層セルにおける DRC エラーを TOP セル上に引き上げて一つの画像で確認できるようにした．そのためには座標系の変換が必要になるがエラー図形ごとに座標変換するのではなく，エラーDB 上のセル単位で TOP セルに重ね合わせている．下位階層セルを重ね合わせるためには，配置点の Angle や Mirror といった情報が必要になるが，それらを TOP セルのセル引用情報から取得している．

また，従来のバッチ処理では難しかったエラー図形を“ある程度のまとまり”で処理するという点に着目し，階層的クラスタリング法を使用することで自動的に適切なクラスタサイズで画像を保存することにした．エラー図形を図形間の距離によりクラスタにしたことで，エラー図形を適度な大きさで確認し，かつ画像の出力数も最低限に抑えることが可能になった．

今後の課題としてはクラスタリングの処理時間があげられる．階層的クラスタリングを用いた場合，総当たりで実行すると時間計算量は $O(n^3)$になり[10]，データ構造やアルゴリズムを工夫しても $O(n^2 logn)$となることが知られている[11]. DRC のエラー図形数は LSI の大規模化や複雑化によってさらに多くなると予想されるので，処理時間短縮が重要である．

参考文献

[1] 名倉徹，LSI設計常識講座，東京大学出版会，東京，(2011)
[2] R. Goering, “Design rule check, layout-vs.-schematic tools roll,” Electronic Engineering Times, 1388, pp.31-47, (2005).
[3] S.M. Rubin, Computer Aids for VLSI Design, 2nd ed., Addison-Wesley Publishing Company, California, (1994).
[4] 宮本貞明，クラスター分析入門，森北肇，森北出版（株），東京，(1999).
[5] Calibre nmDRC/nmLVS 簡易リファレンス Software Version 2008.4, Mentor Graphics.,(2009).
[6] LAVIS-plus.Users-Guide, TOOL Corp., (2014).
[7] lverrdb2gds.txt, TOOL Corp., (2014).
[8] M. Lutz, Programming Python Fourth Edition, O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA95472, (2011).
[9] 佐藤義治，多変量データの分類，朝倉邦造，（株）朝倉書店，東京，(2009).
[10] 石橋徹夫，古賀久志，渡辺俊典，菅原研，“Locality-Sensitive Hashing を用いた階層的クラスタ解析手法の高速化”，情報処理学会研究報告，2003-CVIM-141，pp.57-62，(2003).
[11] A. Bouguettaya, Q. Yu, X. Liu, X. Zhou and A. Song, “Efficient agglomerative hierarchical clustering,” Expert Systems with Applications, 42, pp.2785-2797, (2015).