早稲田大学審査学位論文（博士）

大規模LSIのレイアウト設計検証に関する研究

亀井　智紀

早稲田大学大学院情報生産システム研究科

2016 年 7 月

概要

本論文は，大規模 LSI(Large Scale Integrated Circuit)　　の設計自動化システム（EDA: Electronic Design Automation　）の開発に関するもので，特にレイアウト設計および検証ツールに関する研究をまとめたものである．

近年の LSI 設計技術，微細加工技術および製造技術の目覚ましい発展により，システムオンチップ(System-on-Chip)　のような LSI の大規模化と複雑化が進み，これに伴って設計や製造上の様々な問題が発生している．たとえば，回路・レイアウト設計の設計期間や人的資源の増大，テスト時間やテスト困難性の増大，マスクパターン形状の不具合による製造歩留り低下などがある．また，大規模化によって設計データ量が増大し，ファイルアクセスに長時間を要することやデータ保存場所の資源の圧迫，さらに設計データを外部の製造委託先に送る際のデータ転送時間の問題も起こっている．特にレイアウト設計工程においては，膨大な量の図形データの処理や DRC(Design Rule Check)　とよばれる幾何学的検証が厄介な問題として顕著になってきた．

そこで本研究では，EDAツールの視点からレイアウト設計工程での問題点に着目し，その解決手法を提案する．また有効性を確認するために，提案手法を実際の EDAシステムに組み込んで実験評価を行った．対象とする問題は大きく分けて，(1) 大規模データの DRCツールに関する問題および(2)人手による検証作業に関する問題の 2 点である．これらの問題の背景と求められる解法の要点は次の通りである．

(1)DRC ツールに関する問題

LSI の大規模・複雑化に伴って，レイアウト設計データである図形の数が急激に増加している．中でも，異なる配線層間を電気的に接続するビア(Via) による図形数の増加が顕著になっている．実際の LSI の設計データでは，すべての図形数に対してビア図形数の占める割合が大半であり，たとえば車載用 LSI の例ではレイアウト図形の 99.6%がビア図形というものもある．さらに近年，製造容易化設計（DFM: Design for Manufacturing)　や歩留り考慮設計（DFY: Design for Yield　）の考えが重視され，製造ばらつき対策や信頼性確保の観点からビアを 2 重に設ける冗長ビア（Redundant Via　あるいは Double-cut Via)　が使用されている．これは，回路構造上は新たに並列回路を構

成したことと等価であり，冗長ビアの数に比例して回路構造がさらに複雑となる．したがって，DRCツールを用いて配線幅のチェックを行う際，すべての配線ネットの配線幅をチェックすることになり，膨大な時間を要する．そこで，配線幅チェックの前に冗長ビアをDRCの対象から除外し，処理時間を大幅に減らすことが求められている．また別の問題として，配線のレイアウトにおいて，同一の配線ネット中の一部の配線幅を意図的に細くするテーパリング処理(tapering)　を行い，必要以上に太い配線による面積の無駄を削減することがある．その細い配線部分は設計ミスではなく擬似エラーなので，配線幅チェック後のエラーから手動で取り除いている．しかし，テーパリングが多用される電源配線では，LSIの大規模化により電源配線も長大化し，擬似エラーの数も膨大になるので，人手で取り除くことは困難である．そこで真にチェックすべき幹線のみを取り出し，幹線以外の枝線をチェック対象から除外する手法が求められている．

（2）人手による検証作業に関する問題

次にDRCツールにより検証結果が得られた後の問題について考える．ここ数年，設計および検証作業の水平分業が進み，DRCで検出されたエラー内容を異なる部門の作業者で確認することが多くなった．このため，設計後のDRC出力結果を標準的な画像ファイル(PNG,JPEG,BMP等)の形式で保存している．しかし，これまでのDRCツールの出力は単にエラー図形の座標データであり，エラー箇所とその原因を理解するためには，この座標データを図形化して元のレイアウトデータに重ねて表示する必要がある．重ね合わせはレイアウトデータを構成する基本要素であるセル単位となるが，LSIデータが大規模になればセル数は増加し，さらにエラー図形の数も増えることから，必然的に作業者の負担も増大する．この問題に対処するためには，エラー図形とレイアウトデータの重ね合わせから画像保存までを自動的に行うと同時に，エラー発生の原因を利用者が容易に把握できるように表示するシステムが必要とされている．

上述の2点の問題に対する解決手法は，これまでのEDAシステムには備わっていないか，または同様の機能を果たすために設計者やDRC作業者の大幅な介在が必要であった．本研究では，解決手法を提案し，EDAシステムに実装して効果を確認した．その結果，構築したシステムはLSIのレイアウト設計検証の効率を大幅に向上できることが示された．

以下，本論文の構成を説明する．

第1章「序論」では，本研究の背景と目的ならびにその意義について述べる．特にLSIの大規模化，微細化が進んだことにより幾何学的検証の工程で起こっている問題を具体的に挙げ，本研究の目的を明確にする．

第 2 章「準備」では本研究における実験環境，使用ツールおよびツールの改修について説明する．第 3 章以降の提案手法はすべて既存の EDAツールに実装して実験・評価する．そのために，API(Application Program Interface)　を用いた EDAツールの改修方法を紹介する．また，研究に使用したレイアウトデータのフォーマットとして，標準形式の一つである GDSII(Graphic Data System)　フォーマットを説明する．

第 3 章「レイアウトデータ削減による効率的 DRC」では LSI レイアウト設計における冗長ビアおよび配線のテーパリングを説明し，これらのレイアウトデータが DRC上の障害になっていることを述べる．その上で，基本的な解決手法としてそのようなデータを DRC対象データから削除する方法を提案する．すなわち，冗長ビアについてはレイアウトデータから仮想的に削除し，テーパリングによる設計基準値以下の幅細の配線部分は配線木構造から枝刈りを行う．提案手法を既存の EDAシステムに実装し，実レイアウトデータを用いて実験した結果，大幅な時間短縮効果が得られた．

第 4 章「電源配線幅の高速検証システム」ではレイアウトデータ削減方法を組み込んだ DRC検証のバッチ処理システムの構築について述べている．DRC自体の処理時間の短縮は第 3 章の提案手法で得られるが，DRC対象から削除するレイアウトデータを適切に指示することが必要である．DRC作業者は設計基準書，回路図およびレイアウト図を参照しながら操作するため負担が大きい．また，DRCは一般に長時間を要する処理である．そこで，EDAシステムの機能の一つである LVSを用いて，回路設計データとレイアウト設計データとを自動照合し，DRC対象部分の絞り込みを容易にする方式を提案する．この絞り込み結果をシステムにあらかじめ与えることで，作業者の負担減と高速バッチ処理が同時に実現された．

第 5 章「DRC検証作業の支援システム」では DRCの出力結果を自動的にレイアウトデータに重ね合わせた上で画像保存を行うシステムについて述べる．レイアウトデータはセルを基本とした階層構造なので，上位階層から最下位層のエラー図形を視認できるように画像保存する．これにより，作業者はエラー図形のセルから周囲に伸びている配線などのレイアウト結果を含めて確認することができ，エラーの原因把握が容易になる．また大量にエラー図形が出力された場合の対策として，出力画像数をできるだけ少なくなるために，エラー図形を「ある程度のまとまり」でグルーピングして画像保存を行う手法を提案する．グルーピングのアルゴリズムとしては「階層クラスタリング法」および「単リンク拡張法」とよぶ 2 通りの方式を提案し，エラー図形の多寡によって，処理時間と表示精度および設定の容易さの関係からこれら 2 手法を切り替えできるようなシステムを構築した．

第 6 章「結論及び今後の課題」では本研究で得られた成果をまとめ，今後の課題について述べる．

謝辞

本論文をまとめるにあたり，終始暖かい激励とご指導，ご鞭撻を頂いた早稲田大学大学院情報生産システム科教授　渡邊孝博博士に心より感謝申し上げます．渡邊教授には筆者が同大学院同研究科修士課程在学時より，ご指導を頂きました．後期博士課程においては，社会人学生という指導の難しい状況の中，何度も筆者の勤務する会社に足を運んでいただき，熱心に discussion　していただきました．

学位論文審査において，貴重なご指導とご助言をいただいた早稲田大学大学院情報生産システム科教授の吉村猛博士，同教授の木村晋二博士に心より感謝申し上げます．

株式会社東芝セミコンダクー&ストレージ社の川北真裕氏には LVS の結果を利用した配線幅チェックに関して多くの助言をいただきました．ここに感謝申し上げます．

業務が多忙な折にもかかわらず，大学院入学を許可してくださった TOOL 株式会社の本垰秀明元会長，平井伸治 EDA 技術本部長に厚く御礼申し上げます．また同社開発部の安部拓哉氏には枝刈りアルゴリズム他で多くの知見をいただきました．同社 EDA 製品企画室の高橋利和室長には論文提出の際に多くの助言をいただきました．心より感謝申し上げます．元 TOOL 米国支社増田幸弘氏には英語表記の校正をしていただき大変助かりました．また研究を進めるにあたりご支援，ご協力を頂きながら，ここにお名前を記すことが出来なかった多くの方々に心より感謝申しあげます．

最後になりますが，筆者が TOOL 株式会社入社以来 EDA カスタマイズの方法を熱心に教授してくださり，筆者の大学院入学の際には推薦文も書いてくださりながら平成 25 年 1 月 8 日に永眠した元 TOOL 株式会社 EDA 事業部技術部長の堤康雄氏にご哀悼の意を示すと共に深く御礼申し上げます．

亀井智紀
東京，日本
2016 年 8 月

目次

表一覧

図一覧

第 1 章　序論

1.1　研究の背景

1.1.1　半導体の微細化

LSI における半導体デバイスは年々微細化，高集積化を繰り返しながら今日まで進んできた．1970 年に 10 μmだった半導体の加工寸法は，最新の LSI では 14nm 付近まで小さくなった．現在は微細加工の技術は物理限界，製造・設計限界から大きな壁に当たっているが，2000 年頃まではチップ面積当たりの集積度が 18 か月で 2 倍になるという「ムーアの法則」（Moore’s law ）を体現していた．この微細加工の技術がコンピュータの進化を促し，デジタル家電や携帯電話機といったデバイスを生み出してきた．

図 1.1 に，International Technology Roadmap for Semiconductors(ITRS)[1] が作成した半導体製造技術のロードマップを引用する．おおよそ 2〜3 年でトランジスタゲート長は 0.7 倍，つまり面積が半分になっている．現在，CMOS の限界は 5nm 程度と言われている．しかし，これまで唱えられてきた CMOS の微細化限界論は，新たなデバイスの研究や新技術の投入により延命し続けている．微細化の限界を正確に予測することは困難だが，ITRS では 2022 年に 4.5nm という予測を出している．

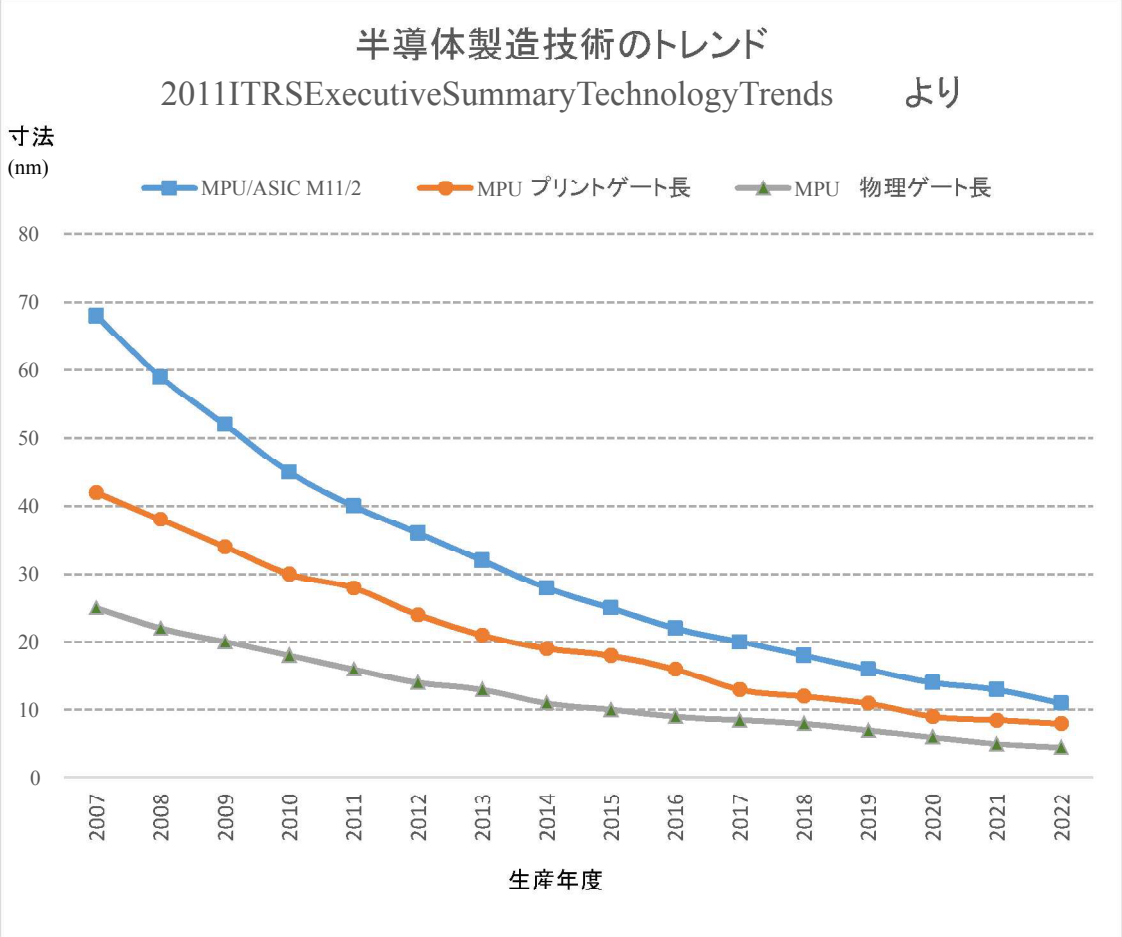


図 1.1 半導体生産技術のトレンド

ITRS 2011 Edition Executive Summary から引用

1.1.2 LSI 設計・製造手順と EDA ツール

LSI の代表的な設計・製造手順として，一般的な特定用途向け集積回路(ASIC: Application Specific Integrated Circuit) を対象とした例を図 1.2 に示す．設計工程上の各段階でコンピュータ上の EDA(Electronic Design Automation) ツール[2] が使用され，論理シミュレーション，回路解析[3]，デバイス解析，配置・配線設計[4]，マスクパターン作成[5]，テストパターン作成[6] などの作業が進められる．今日，システム LSI のように大規模化した集積回路の設計には EDA ツールを使用することなく設計することは不可能である．本研究ではレイアウトデータ解析の EDA ツールとして LAVIS を使用したが，その詳細については後述する．

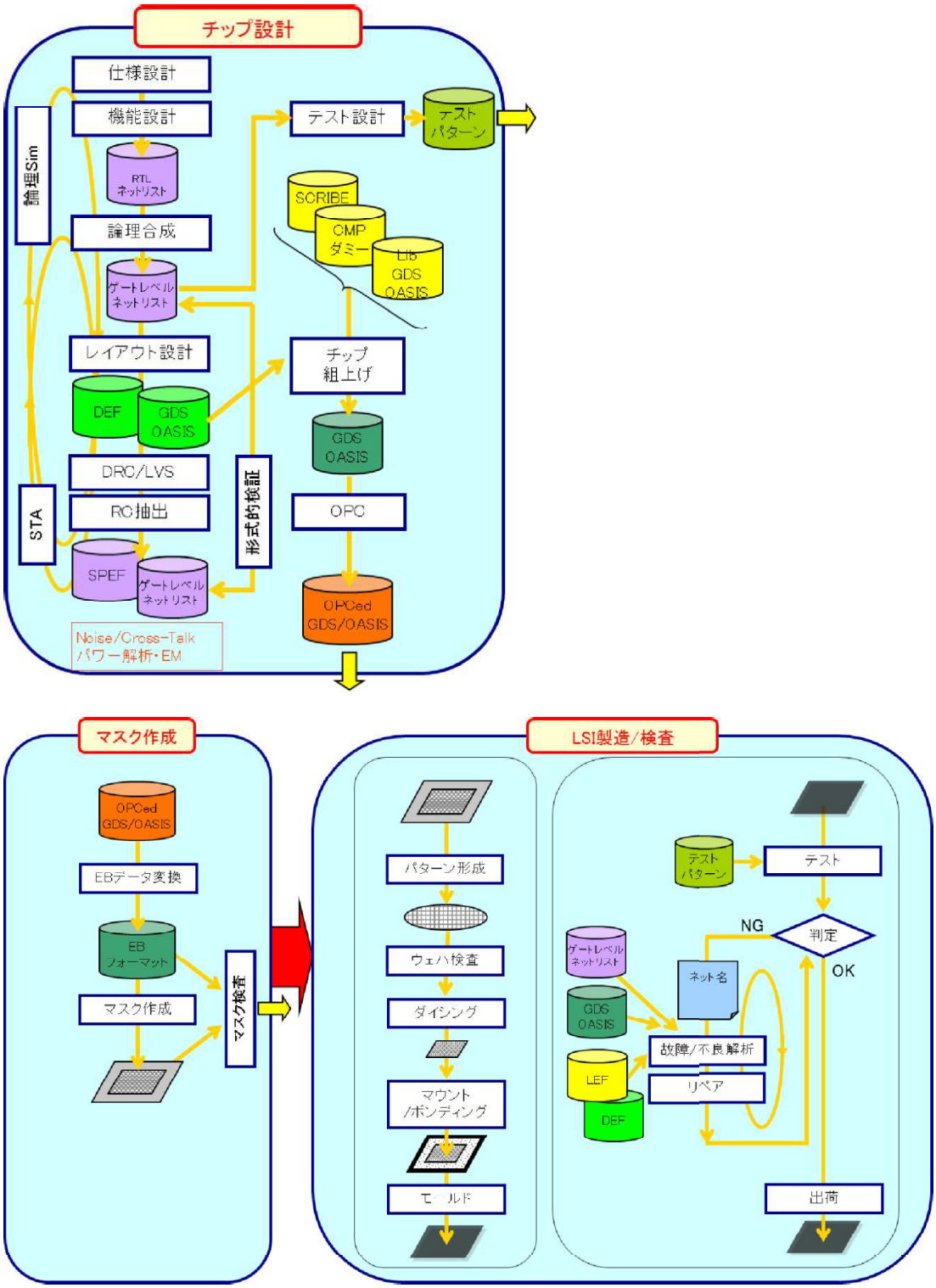


図 1.2　LSI 設計・製造手順

Fig.2 LSI Design and Manufacturing process

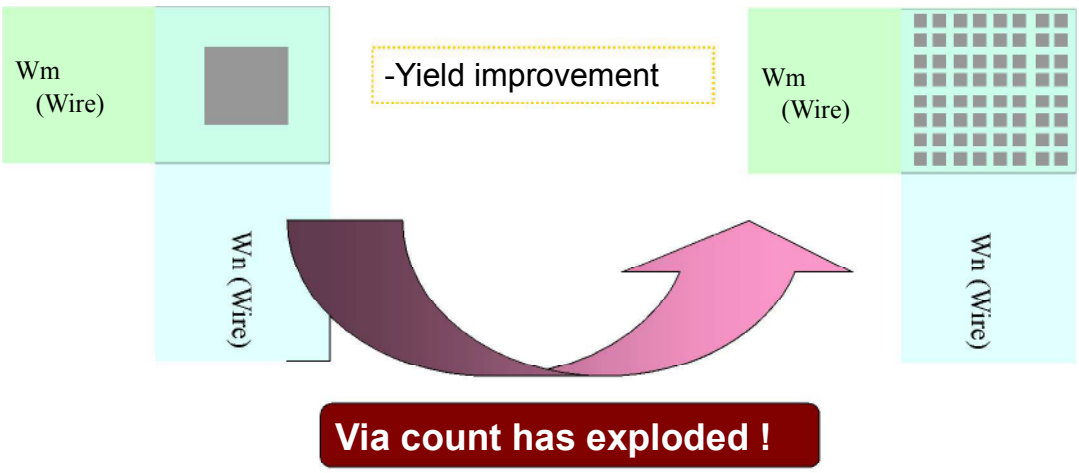
1.1.3 半導体微細化に伴うEDAの問題

1.1.1 および 1.1.2 で半導体微細化およびEDA ツールの役割について述べた. 特にレイアウト設計工程においては半導体微細化の傾向により，膨大な量の図形データの処理や図形の幾何学的検証が厄介な問題として顕著になってきた．根幹は GDSII に代表されるレイアウトデータの図形数が増え，データファイルのサイズが増加していることにある．以下に本研究で考える問題の個別例を挙げる．

(1) ビア図形増加による解析時間増大の問題

LSI の大規模・複雑化に伴って，レイアウト設計データである図形の数が増加することになるが, 中でもビア図形の増加が顕著になっている. 昨今, 製造容易化設計（DFM:Design for Manufacturing ）[7][8] や歩留まり考慮設計（DFY: Design for Yield)[9] の考えが普及し，製造ばらつき対策や信頼性確保の面から異なる配線層を接続するビアを 2 重に設ける冗長ビア（Redundant Via あるいは Double-cut Via) が使用されている[10][11] ．例えば，実際のVLSIのチップで, 全ポリゴン数が約2570 万に対して, ビア数が約2560 万であり, 約99.6 %がビアとなっている例もある．(図 1.3)

ビア図形の増加はレイアウトの解析，検証を行う EDAツールにとっては，極めて大きな問題となる．EDAツールは，配線幅チェック等の設計基準検査（DRC: Design Rule Check ）[12] を実行する際，層間の接続を追跡するためにビア 1 個で接続される箇所が一つの回路となる．ゆえにビア個数分の並列回路の経路を解析することになり，ビアが大量に存在すると計算コストが増大する．このため，EDAツールを使用した通常の手法の DRCでは処理に時間がかかる上に，大量のメモリ空間を必要とするため，LSI 検証工程の大きな問題になってきている(図 1.4) ．抵抗値計算や配線幅チェックにおいて，より顕著にこの現象が見られる．



Ex) In a GDSII data, 99.6% of total polygon belong to Via

図 1.3　DFM の副作用

Fig 1.3 Side effect of DFM/DFY

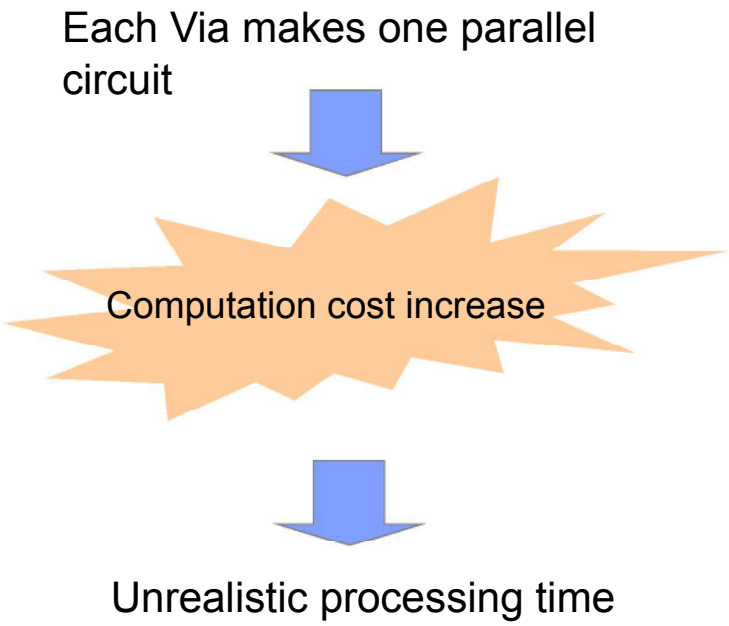


図 1.4 EDA ツールで解析・検証を行う上での問題点

Fig 1.4 Problems on EDA tool analysis/verification

(2)　電源配線の長大化による DRCの問題

チップ面積あたりのトランジスタ数が増加したことにより，引き回す電源配線が長大化している．電源配線の幅が細すぎると過剰な電流が流れた場合，溶断事故を引き起こすため DRCツールを使用して検証を行う必要がある．しかし，アナログ回路の LSI に多くみられる電源ネット[13] では，配線が分岐するたびに配線面積や電気的性質を考慮して配線幅を調節する処理が行われる[14] ．このため，同一ネットでありながら配線幅が細くなっている部分があるため，電源ネットすべてを対象にした配線幅検証ではこの部分が擬似エラーとして大量に検出されるという問題がある．(図 1.5)

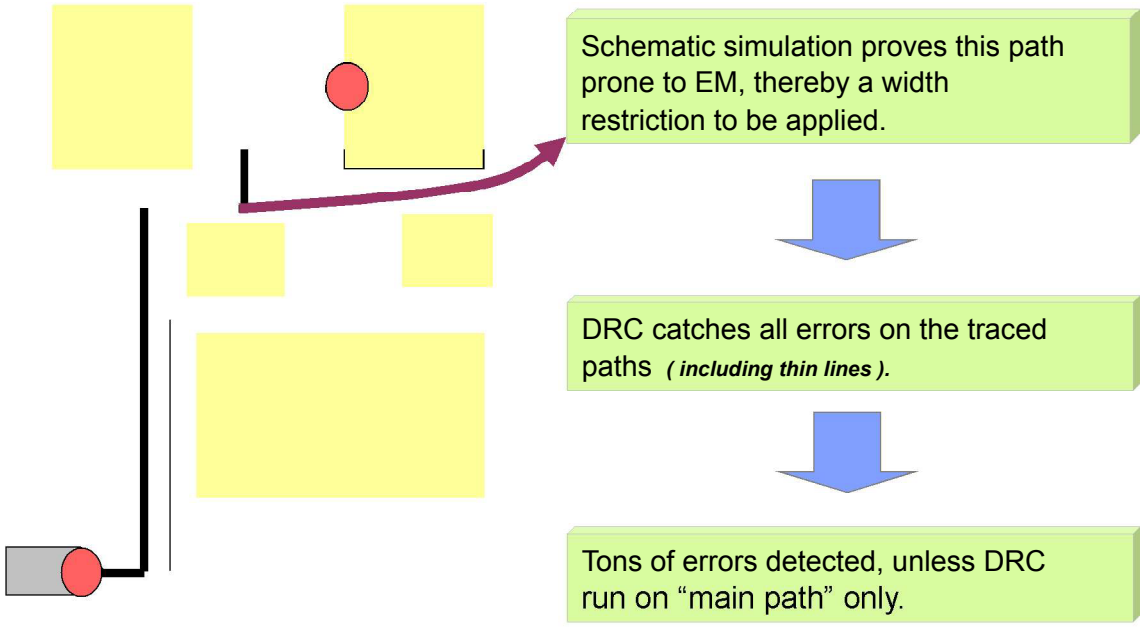


図 1.5 電源配線と DRC

Fig.1.5 PG-Wire and Design Rule Check

(3)検証箇所の増加による人間の作業時間増大の問題

電源配線が長大化したため，検証を行わなくてはいけない個所も増加する．通常，回路設計者とレイアウト設計者は異なる．先ず，回路設計者が回路図上で検証が必要とした個所をレイアウト設計者は設計したレイアウトデータ上から探し出し，配線幅等が設計規約に合っているか検証を行う．検証箇所が増えたことにより，この作業も莫大な時間を要している．

(4)DRC 結果の画像保存に伴う作業者の負担の増大

(3)で述べたように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，レイアウトデータが大きくなることにより，エラー図形は大量になる傾向があり，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る．

1.2 研究の目的

以上に述べた状況を踏まえ，本研究では，レイアウトデータの大規模化に対応したEDAツールを開発するため，以下の三点に着目した研究を行った．このうち一点目と二点目はレイアウトデータを削減する処理に関するもので，三点目は画像保存と表示に関するものである．

まず一点目は解析時における冗長ビアの削減である．もっとも実際のレイアウトデータ上で削減しては意味がないので，解析時のデータベース上での削減を試みた．さらに電源ネットから基幹線のみを抽出し，支線の枝を刈ったうえで 2 点間の配線幅チェックを行うシステムを提案し，効果を評価することである．

次に二点目は配線幅チェックの完全自動化システムを構築することである．レイアウトデータ上の 2 点間の座標を算出するためにレイアウト・回路照合システム（LVS:Layout Versus Schematic ）[15] の結果を利用するシステムを提案する．

最後に三点目はDRC 結果を自動的にレイアウトデータに重ね合わせた上で，その画像を保存するシステムの作成である．重ね合わせる上ではセルの階層を考慮しなければならない．さらに，出力画像を大量に生成しないように，DRC 結果のエラー図形を「ある程度のまとまり」で画像に出力する工夫をした．

第 2 章　準備

2.1　実験環境

本研究の EDA ツールの開発と実験では，その内容に応じて異なる実験環境が必要であったため，表 2.1 に示す 2 種類の環境を利用した．表では代表的なマシンスペックのみを記している．

表 2.1　実験環境

Table 2.1 Experimental environment.

マシン１

CPU	4 x AMD Opteron 850 (2.4GHz)
Memory	64GB
OS	CentOS4 (x86_64)

マシン２

CPU	8 x AMD Opteron 6128 (2.0GHz)
Memory	32GB
OS	CentOS5 (x86_64)

2.2　実験データ

本研究において使用しているデータフォーマットは GDSII　形式である．GDSII　は前身の GDS　が米国 Calma　社の CAD　ツール用に開発されたレイアウトデータのフォーマットで，GDSII　はその後継にあたる．現在，半導体業界でレイアウトデータとして最もポピュラーに使用されている形式である[16][17]．Stream　フォーマットとも呼ばれている．一般にレイアウトデータは矩形や多角形といった図形要素と，それらを機能ブロックとしてまとめたセルから構成されている．セルは他のセルから参照することも可能で，その際はアレイとしても表現できる．

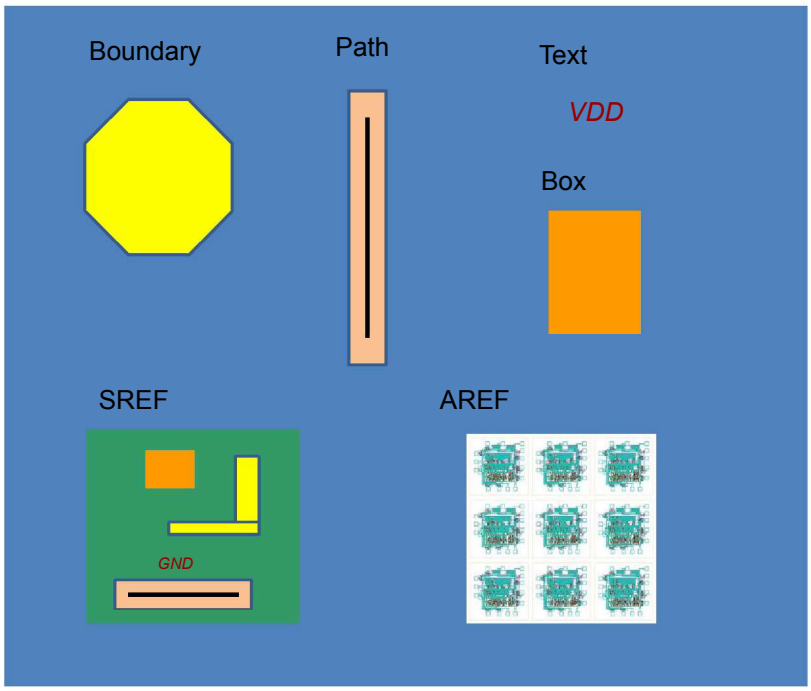


図 2.1　GDSII　エレメントタイプ

Fig.2.1 GDSII element type

2.3 実験システムフレームワーク

本研究では EDA ツール LAVIS[18] （LAVIS-plus[19] ）に提案手法を組み込み，実験・評価を行った．LAVIS （LAVIS-plus ）は TOOL Corporation の製品である．当初は GDSII や OASIS[20][21] といったレイアウトデータを表示する Viewer として開発されたが，現在では等電位追跡（NodeTrace ），抵抗値計算，DRC などの機能も備えており，レイアウトデータの検証ツールという位置づけになっている．LAVIS およびその後継である LAVIS-plus は商用の製品であるが，後述する API[13] を使うことにより機能の追加や変更といったカスタマイズを行うことができる．本研究のほとんどはその手法により開発したものだが，一部，LAVIS （LAVIS-plus ）の本体に改修を加えたものもある．

LAVIS-plus は LAVIS の後継であり，本研究の途中の過程で置き換えが始まった．本論文中では第 5 章で LAVIS-plus を使用している．それ以前の章では LAVIS を使用している．LAVIS と LAVIS-plus では API の仕組みが大きく異なっており，機能をカスタマイズする手順も差異がある．以下ではまず API について説明し，その後，LAVIS および LAVIS-plus の改修について述べる．

2.3.1 EDA ツールと API

EDA ツールの多くは API の仕組みを持ち，ユーザの利便性を図っている．API とは，EDA ツールの機能をユーザが簡単にプログラムできるように設定されたインターフェースのことである．ユーザは EDA ツールに実装された API を使用することにより，機能の拡張やそれぞれの仕様要求に応じたカスタマイズを行うことができる．しかし，あらかじめ EDA ツールに用意されている API は機能が限定されているものが多く，API だけではユーザ所望のカスタマイズができない場面も発生する．一部の EDA ツールには，Python[22] や Tcl[23] あるいは Ruby[24] といったスクリプト言語に対応して，ユーザがプログラミングできる仕組みになっているものもある．しかし，本研究で使用した LAVIS は，既存のスクリプト言語を利用してプログラミングする仕組みを持っていない．この種の EDA ツールにおいては，あらかじめ EDA ツールベンダーが用意したコマンド群[25] を駆使してプログラミングを行うのが一般的である．

2.3.2.2 フレームワークの改修手法

EDA ツールベンダーが用意した API だけでは所望の機能が実現できないため，次のような方法を採用した．

(1) API から OS に標準実装されているスクリプト言語を呼び出し，そちらで処理を実行する

(2) OS のスクリプト言語で処理した結果を，API に戻す

(1)に関して述べる．通常，EDA ツールは OS のシェルにパラメータを直接渡す API を実装している．（LAVIS の場合は”system ”というコマンド）このような API を使用して，Unix 系の OS に実装されている Perl や Python といったスクリプト言語を起動する．スクリプト言語側に渡したいデータがあれば，引数渡しやファイル渡しができる．この方法は論文[26]にも紹介されており，EDA ツールのカスタマイズをする際に一般的に行われている．

次に(2)について述べる．(1)が実行され，OS のスクリプト言語により何らかの処理が行われたとする．この処理結果を API を介して EDA ツールに戻すために，以下のことを行う．

・スクリプト言語から API に返したいデータは，API の文法でスクリプト言語がテキストファイルに出力する

・(1)が終了し，API に処理が戻ったときに API 側でこのテキストファイルをサブルーチンとして実行する

・結果的にスクリプト言語の処理を API で取り込むことができる．

以下に例を示す．

LAVIS には平方根を求める API は用意されていない．そこで上記の方法を使って平方根を求める．OS に実装されているスクリプト言語の例として Perl を使用する．先ず EDA ツールが Perl モジュールを呼び出す．system は OS システムコールを起動する API である．呼び出された Perl では平方根の計算をして，その結果を LAVIS の API が解釈できる文法によりファイルに出力する．API では Perl から出力されたファイルをサブモジュールとして，プログラムを進行する．source はサブルーチンをコールする API である．このようにして EDA ツールで提供されている API コマンドでは計算できなかった平方根の値を，EDA ツール側で取得できる．フローを図 2.2 に示す．

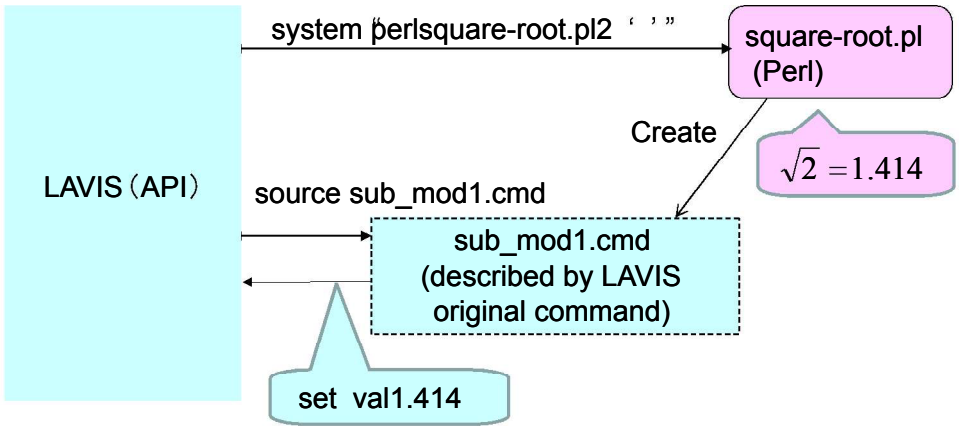


図 2.2　EDA ツールの API とスクリプト言語

Fig.2.2　　API of EDA tool and script language

本研究では 3 章で紹介するビアの削減，電源ネットの枝刈り処理および 4 章では，すべて上記の手法を駆使して，EDA ツールの改修を行った．この手法については LAVIS 以外の EDA ツールでも適用でき，既存のスクリプト言語を利用してプログラミングが行えるタイプの EDA ツールにおいても，別の言語をコールすることにより，特定のライブラリ資産が使える等のメリットがある．

ちなみにビアの削減と枝刈り処理は LAVIS 本体の機能を改修して実装した．LAVIS-plus を使用した 5 章では後述する手法を使っている．

次に，LAVIS-plus の場合について述べる．LAVIS-plus では API として python 言語をサポートしている．したがって，改修作業は LAVIS より簡略化される．先の平方根を求める例で考えると，LAVIS-plus も LAVIS と同様に平方根を求める API というものは提供されていない．しかし，LAVIS-plus の場合は python を解釈できるので，

```
import math
val = math.sqrt(2)
```

とプログラムを書くだけでよい．

第 3 章　レイアウトデータ削減による DRC の効率化

3.1　はじめに

近年, GDS Ⅱに代表される LSI レイアウトデータの総サイズは, さらに増加傾向にあり, ギガバイト超のデータも珍しくない．レイアウトデータの中でも電源線の箇所は，十分な配線幅が確保されていないと，過剰な電流に対して溶断事故が発生するため，DRC の重要性は極めて大きい．しかし，ギガバイト超のレイアウトデータの電源線ネットは長大かつ，それを構成している図形数も数百万のオーダになるため，DRC 処理時間が深刻な問題となっている．

一般的に大電流が流れるネットでは層間接続に非常に多くのビアが使用され[27]，さらに DFM や DFY の一つとして，冗長ビアを付加する手法も提案されている[10][11]．これは，製造ばらつき対策や信頼性確保のために有効であるが，結果として多層配線において配線間を接続するビアの数が著しく増加し，レイアウトデータのサイズをさらに肥大化させている．例えば，実際の LSI のチップで，全ポリゴン数が約 2570 万に対して，ビア数が約 2560 万であり，約 99.6 %がビアとなっている例もある．

ビア図形の増加はレイアウトの解析，検証を行う EDA ツールにとっては，極めて大きな問題となる．EDA ツールは，電源ネット等の配線幅チェック等の DRC を実行する際，層間の接続を追跡するためにビア 1 個で接続される箇所が一つの回路となる．ゆえにビア個数分の並列回路の経路を解析することになり，ビアが大量に存在すると計算コストが増大する. このため, EDA ツールを使用した通常の手法の DRC では処理に時間がかかる上に，大量のメモリ空間を必要とするため，LSI 検証工程の大きな問題になってきている．

一方で，例えば電源ネットの配線幅チェックを行うのであれば，先ず多層に渡る電源ネットを抽出するために，電気的な配線間の接続を示すための情報を持つ最低限のビア情報が必要である．しかし，そのためには一接続箇所当たり一個のビアが存在すればよく，層の接続箇所での大量のビア一個一個を処理する必要はない．そこで，本研究では先ず DRC

処理には冗長なビアを電源ネットから取り除き，得られたネットに対して，配線幅チェックを行う手法を提案する．また，2 点間の経路上でネットの枝刈りを行う手法についても提案する．それらを EDA ツールに実装し，効果を確認する．

実験の結果，電源一ネットあたりの総図形数は大幅に減少し，ネット全体に対しての配線幅チェックは瞬時に結果を得られるようになった．さらに，ネット中の 2 点を始点，終点として指定し，指定経路に沿った配線幅チェックを行う処理についても実装を行った．

3.2 冗長ビアの削減

3.2.1 ビア削減手法

ビアは全て配線図形内に包含されているものとする．今回提案する手法は，配線幅チェックを実行する前段階で配線層の重なっている領域を抽出し，その領域内に一個以上のビアが配置されていたら，その領域全体を一個のビアとした上で，元々存在しているビア情報を DRC 対象データからすべて破棄する．本手法により，一接続箇所あたりのビア数は 1 になるため，大幅に図形数を削減できる．ただし元データに存在した複数のビアの幅チェックは行えなくなるため，その場合は削減を行っていないデータを対象にビアのレイヤのみで配線幅チェックを実行する．

DRC を行うツールは商用の TOOL 株式会社製 LAVIS を使用する．図 3.1 に本処理の全体のフローを示す．大きく，等電位追跡処理と配線幅チェック処理に分かれる．本研究の対象は GDS Ⅱデータなので，図形情報のみでネットリスト[28] は存在しない．そこで先ず，等電位追跡を行い，検証を行う電源ネットを抽出する．この際，どのレイヤとどのレイヤがどのビアで接続される，というテクノロジー定義情報[29] をあらかじめ利用者が作成し，その情報を元に抽出する．このネットを抽出する過程において，上記の手法によりビアを削減したネットを作成する．すると配線層間の電気的な接続は維持された状態でビアだけが削減されたネットができる．配線幅チェックはこのネットに対して実行する．ところで，電源ネットは長大であり，抽出自体に相当の時間を要するため，あらかじめ全ネットのデータベースを作成する作業を行う．一度，データベースを作成しておくと，その後は GDS Ⅱデータに存在する任意の等電位ネットを瞬時に抽出することができる．このデータベース作成段階で，以下のようなビアの削減処理を行う．

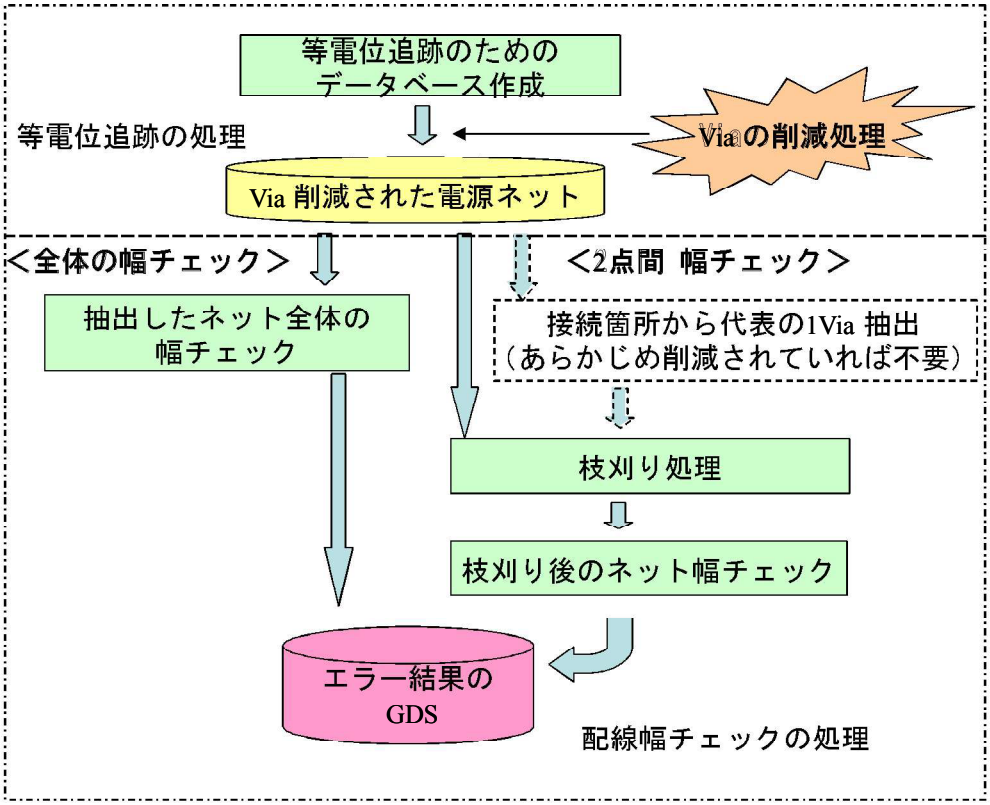


図 3.1　等電位追跡から配線幅チェックまでの処理フロー

Fig.3.1 Entire processing flow of the wire width checking

例えば, Metal1 配線と Metal2 配線がビア 12 で接続されている例を考える. 先ず, Metal1 配線の図形と Metal2 配線の図形を抽出し，双方が重なっている AND 領域を取り出す．その AND 領域内に一個以上のビアが存在すれば，領域全体を一個のビアとして配置，元のビアは破棄する．この手法を式(1)，(2) および図 3.2 に示す．ここで，配線レイヤ m，n の配線図形の領域を Wm，Wn とし，ビア図形の領域を Vmn とする．また，∩，⊃はそれぞれ図形演算の AND，包含を表す．テクノロジー定義情報に定義されている全ての配線レイヤとビアレイヤについて，同様の処理を行う．

$$(W_m \cap W_n) \supset V_{mn} \qquad (1)$$

$$V_{mn} \Leftarrow W_m \cap W_n \qquad (2)$$

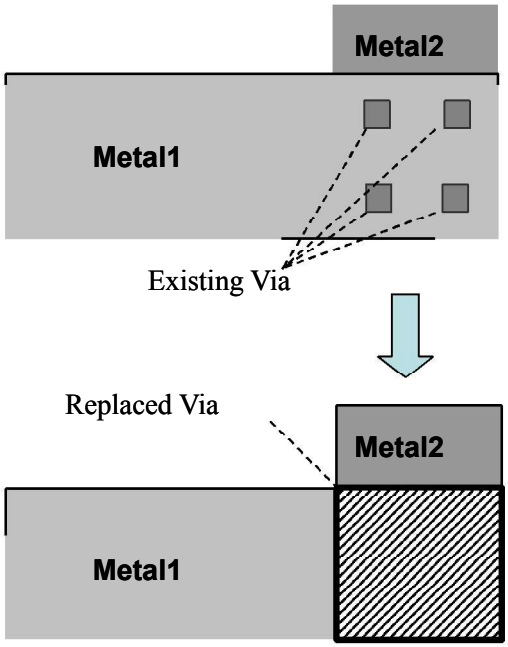


図 3.2　ビアの削減手法

Fig 3.2 Via reduction method

3.2.2　実験（LAVIS）

（1）データベース作成時間

実験には表 3.1 のデータを使用する．電源ネットを抽出しながら，ビアの削減処理を実施したデータベースの作成時間を表 3.2 に示す．データベース作成の際には，CPU 分散が可能である[30][31]．今回，実験に使用したマシンは 4CPU を搭載しているので，4CPU 分散処理でデータベースを作成した．ビアの削減処理を実施したものは，しないものと比較して 2 分半ほど時間がかかった．ただし，データベース作成は最初に 1 回行えばよいだけなので，後続の DRC 処理を繰り返す場合に効率が向上する．

また，データベースのファイルサイズ自体も，ビア数が減少したことにより小さくなる．そのため，データベース読込時間は短縮され，読込時に使用するメモリも少なくなる．

表 3.1　実験に使用したデータ

Table 3.1Experimental data

ファイルフォーマット	GDSII
ファイルサイズ [MB]	1,700
処理領域（設計値） [um]	6,000 x 6,000
配線層数	Metal 4layer + Via 3layer + Poly
プロセス（設計値） [um]	0.2
チップ面積 [um ^{2}]	36,602,550
総図形数	25,876,841

表 3.2 ビア削除有無によっての DB 作成および読み込み

Table 3.2DB creation and reading with or without Via reducing

	Via 削減なし	Via 削減あり
DB作成時間(4CPU分散）[sec]	1055	1217
DB読込時間 [sec]	45	0.003
DBファイルサイズ [MB]	559	5.6
DB読込使用メモリ [MB]	454	23

表 3.3　図形数と幅チェック時間（時間は CPU 時間）

Table 3.3　Number of figuresand width checking time (CPU time)

	Via図形数		処理時間（全体）[sec]	
ネット名	Original	Via削減	Original	Via削減
A	2176634	208	36	<1
B	2377162	2204	36	<1
C	6142784	2122	94	<1
D	9599292	2001	90	<1

（2） 図形数と幅チェック時間

ネットあたりの図形数は，ビアの削減処理を実施したことで，大幅に少なくなる．ネットごとの図形数の変化を表 3.3 に示す.

また，ネット全体の接続箇所数を N，i 番目の接続箇所でのビア数を ci，ネット全体の配線図形ポリゴン数を W とすると，ネットの総図形数 Fcount は以下の式で表すことができる．配線幅チェック時間の計測は Linux の Top コマンドの表示結果を参照したが，結果が

瞬時に得られたものは実測値測定が不可能なので，表 3.3 では<1 とした．

ビアの削減処理をしない場合

$$Fcount = \sum W + \sum_{i=1}^{N} ci \qquad (3)$$

ビアの削減処理をした場合

$$Fcount = \sum W + N \qquad (4)$$

3.2.3 実験（Calibre）

3.2.1 で提案した手法が特定の DRC ツールだけでなく他の DRC ツールでも有効であることを示すために，別の代表的 DRC ツールである Calibre [32] を使用して実験を行った．階層ライセンス（DRC-H ）は使用していない．CentOS5[33] 以上でないと動作しないため，表 2.1 のマシン 2 を使用した．

ここで以下のような実験を行った．

(1) 表 3.3 の A から D のネットを GDSII 形式に変換する（A.gds,B.gds,C.gds,D.gds ）

(2) GDSII データ上で配線間の論理演算（AND ）を行い，出力層を新たなビア層とし，それまで存在したビア層は捨てたデータを新たに作成する（A_rv.gds, B_rv.gds, C_rv.gds, D_rv.gds ）

(3) (1) ，(2) で作成したデータに対し，Calibre で配線幅チェックを行い，要した時間を計測する．

結果を表 3.4 に示す．この場合は LAVIS での実験と同様に，ビア削減によって大幅な処理時間の短縮がなされていた．

表 3.4 ビア削減有無による Calibre での配線幅チェック時間（sec）

Table 3.4 Width checking time (sec) with or without Via reducing inCalibre

ネット名	Original(X.gds)	ビア削減(X_rv.gds)
A	7	<1
B	7	<1
C	25	<1
D	38	<1

3.2.4.4 考察

配線幅チェックの前段階でビアを削減したものは，瞬時に実行結果を得ることができた．接続箇所あたりのビア図形が一つにまとまったことで，全体の図形数が大幅に減ったためと考えられる．ところで，LAVIS の場合，ビア削減処理の有無で幅チェック時間を比較する場合は，データベース作成と読み込みに要する時間の差分を考慮しなければならない．今回，データベース作成において，ビア削減処理を実施すると 2 分半ほど余分に時間がかかる．逆に読み込みにおいては，45 秒短縮される．ビア削減処理なしでデータベースを作成すると，配線幅チェックにかかる時間は最も長いもので 1 分半ほどであるため，複数箇所を検証する場合は，ビア削減を実施したほうが効率が良いといえる．

Calibre での実験においても前処理でビア削減を行うと高速に DRC ができた．Calibre の場合はツール自体を改修することはできないが，今回手動で実行した 3.2.3(1)(2) の前処理部分をツール内部に組み込むことができれば，同様に高速化できることが分かった．したがって，提案手法はツールに依存せず有効であると言える．

なお，表 3.3 （LAVIS ）の Original 処理時間と表 3.5 （Calibre ）の Original 処理時間の差は第一に実行マシンの差（LAVIS は表 2.1 のマシン 1 を使用，Calibre は表 2.2 のマシン 2 を使用，ただしこの 2 台にマシンに大きな性能差はない），第二にツールの結果表示の方法（LAVIS は GUI ，Calibre は CUI ）の差，第三にツール自体のパフォーマンスの差である．配線幅チェックにおいては元々，Calibre のほうが LAVIS より高速である．

3.3　電源ネット枝刈り

次にビアが削減された電源ネットに対し，枝刈りを行った上で配線幅チェックを実施する．枝刈りは開始点，終了点の 2 点の座標を指定して，その 2 点間の経路上をチェックするものである（図 3.3）．2 点指定のものは，配線幅チェック対象が 2 点の経路上に限定されるため，利用者にとってチェック不要の配線を除外することができ，擬似エラーを大幅に削減できる．特にアナログの Electro Migration （EM ）検証[34] においては,SPICE シミュレーション[35] の結果から計測ポイントが絞られている場合が多いため有効である．経路上，並列回路になる場合は，並列部分も全てチェック対象となる．2 点間の経路のみを抽出するために，経路から枝分かれしている配線部分を枝刈り処理によって取り除く必要がある.最終的に，配線エラーが見つかった場合は，エラー部分の図形が別 GDS Ⅱファイルに出力される．

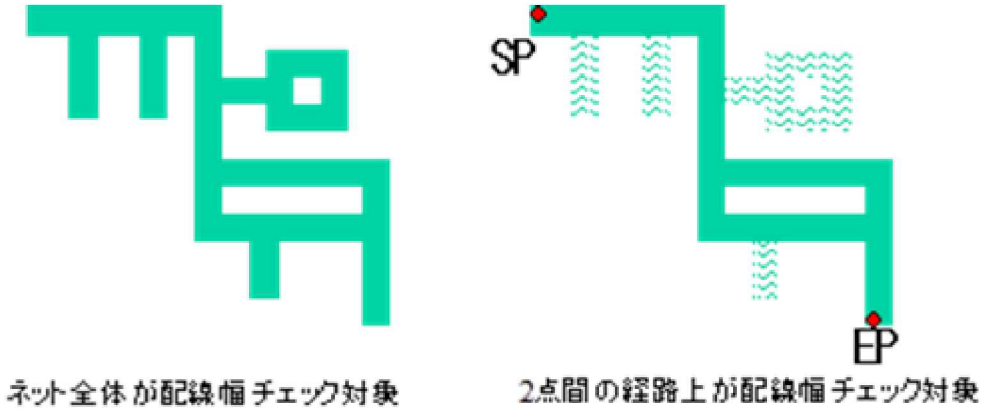


図 3.3 電源ネットの 2 種類の配線幅チェック方法

Fig 3.3Two types of wire width checking method of power net.

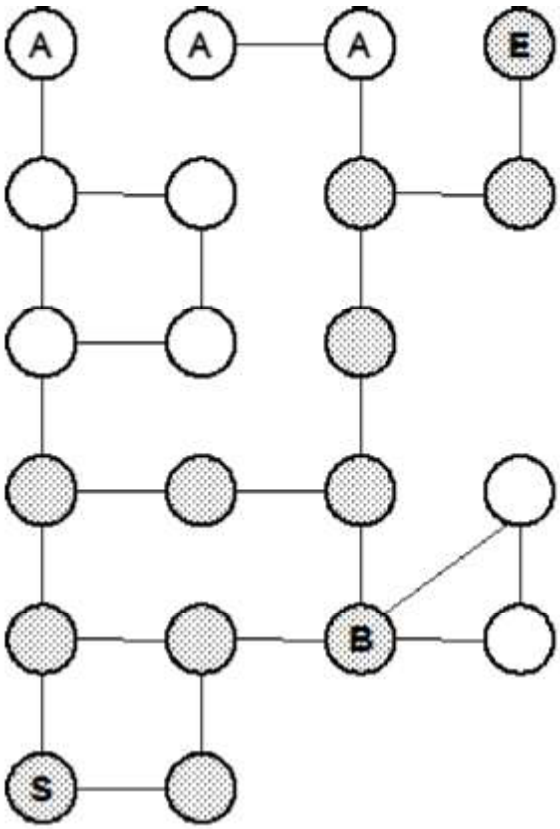


図 3.4　2 点間の枝刈りアルゴリズム

Fig 3.4 Algorithm of pruning between 2 points.

3.3311　枝刈りの手法

例図として等電位ネットをグラフ化した図 3.4 を用いる．このグラフにおいて，エッジは電源-グランド配線であり，S,E を除く各ノードは配線間を接続するビアを表している．

(step.1) S,E 以外の次数 1 のノードとその接続エッジを削除し，グラフを更新する．更新したグラフに新たに次数 1 のノードがあれば，同様に削除しグラフを更新する．図 3.4 では A のノードおよびその接続エッジが削除される．

(step.2) step.1 の結果で得られたグラフを G' とする．G' では S と E の次数は 1 以上であり，それ以外のすべてのノードの次数は 2 以上である．そこで，S,E 間のすべての電流路を配線幅検証の対象として抽出する．

(2-1)　グラフ G' を 2 連結成分に分解したグラフ H を生成する[36]．ここで，2 連結成分

間の連結は関節点(articulation node)　　または橋(bridge)　で保たれている．

(2-2)　グラフ H上で，G’の S を含む 2 連結成分と E を含む 2 連結成分間のパス P を抽出する．

(2-3)　パス P 上のすべての 2 連結成分に含まれる G’のノードから，G’の誘導部分グラフ G’’（図 3.4 のハッチングされたノードおよびそれらを結ぶエッジ）を抽出する[37] ．この結果，パス P 上にない連結成分に含まれる G’のノードとそれに接続するエッジがすべて削除される，なお，P 上にある連結成分との関節点(図 3.4 の B)は削除されないことに注意．

(step.3) G　’’が S-E 間の枝刈りされた経路であり，配線幅検証の対象となるすべての電流パスである．

このアルゴリズムは基本的にグラフの 2 連結成分分解処理と単純グラフの最短パス探索処理であり，時間計算量は頂点数＋エッジ数のオーダになる[38] ．

3.3.2.2　電源ネット枝刈りの効果

従来の方法のように電源ネットすべてを測定の対象とした場合，テーパリングによって配線幅を意図的に細くしている部分も擬似エラーとして検出されてしまい，それらは人手により判断され除外されていた．そして，その箇所が数千個に及ぶことも珍しくないため，作業者の大きな負担となっていた．今回提案した始点と終点を指定して必要な経路のみをチェックする方法により，従来の DRC ツール（代表的な DRC ツールとして例えば[39] ）ではできなかった「作業者がチェックしたい経路のみ検証対象にする」ということが可能になった．

3.4　ネット枝刈りを考慮したビア削減手法の検討

3.3. 節で提案したネット枝刈りを踏まえて，3.2 節でのビア削減手法が適切か検証する．ネット枝刈りを実行して配線幅違反が検出された場合，その配線図形を出力する．しかし，図 3.5 のように Metal1　で配線幅違反が検出された場合，枝刈りをして出力する必要があるため Metal1　の図形をそのまま出力するのではなく，どこかで分断しなくてはならない．最も処理が高速かつ出力図形が不自然な形状にならない方法は接続ビアの重心位置で図形を分断する方法と考え，それを採用した．ここでビア削減を実行した場合，3.2.1 節で提案している配線 AND 領域を 1 個のビアに置き換える方法では，常に配線 AND 領域の中心で図形が分断されるため，分断箇所は一定でエラー図形はどの部分においても同じルールにより出力される．

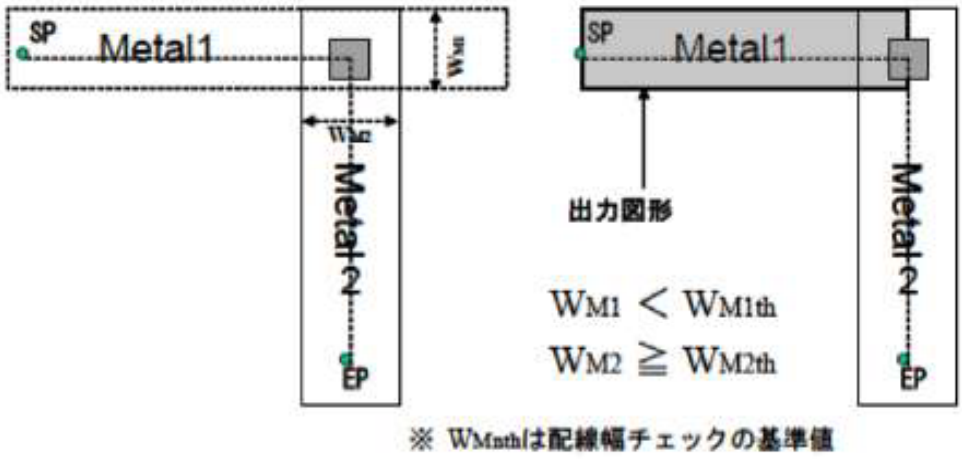


図 3.5　ビアの重心位置でエラー図形を分断

Fig.3.5　An error figure is cut off at the gravity of Via.

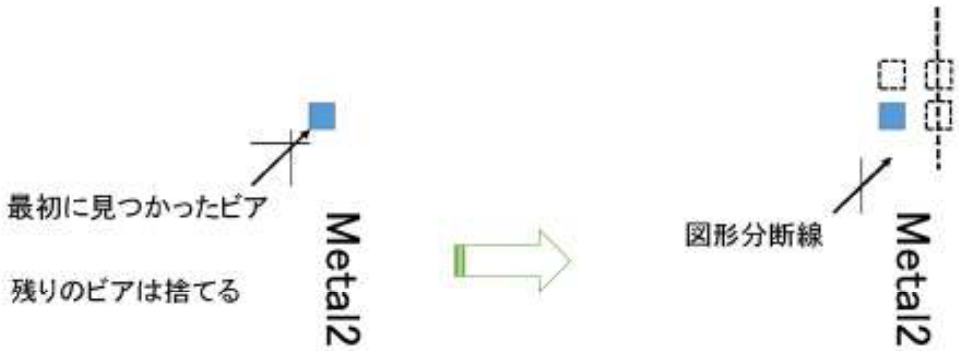


図 3.6　最初に見つかったビアのみ採用する手法と図形分断線

Fig.3.6　Themethodonly using Via found first and figure cut line

一方，ビア削減の別手法として，3.2 節のように配線 AND 領域を 1 ビアとするのではなく，最初に見つかった 1 ビアを採用し残りのビアは捨てるという手法も考えられる．等電位追跡を行う EDA ツールにおいては，この手法を採用していると思われるものも見受けられる．ただ，この手法をネット枝刈りに適用した場合，ビア群のどの位置にあるものが最

初に見つかるかは不定であり図形分断箇所が配線箇所により一定にならないという問題が発生する．図 3.6 では４つのビアのうち左下が最初に見つかったものとし，そのビアのみを残している．そのときの図形分断線は実線の位置になる．しかし，右上もしくは右下が処理上最初に見つかった場合は，図形分断線は破線の位置になり，図形の分断箇所が異なる．この問題を回避するために，例えば，必ずビア群の左下にあるビアを採用するなどとした場合は，ビア群の中から左下を探す処理が必要になりコストが増大する．また，そもそも必ず矩形の形状にビア群が配置されているとも限らない．そこで，ネット枝刈りを実行する際は 3.2.1 節で提案したビア削減手法が最も適切である．

3.5 結論

配線間の接続箇所に大量に存在するビアを一つにまとめ，配線幅チェックに要する時間を検証した．配線間の接続箇所に大量のビアが存在するとネットの配線幅チェック等の DRC に多くの時間がかかっていたためである．大量ビアを一つにまとめる手法として，配線層の交差する AND 領域（図形の AND 演算によって得られる）にビアが一つ以上存在すれば，その AND 領域全体を仮想的に新たなビアとみなす方法を採用した．

ビアの削減処理は配線幅チェックを実行する前段階でデータベース化しておき，以降の配線幅チェックは，そのデータベースを用いて行う．

先ず，ビアの削減処理を行うことで，このデータベースのサイズが減少し，読み込みに要する時間短縮と使用メモリの減少が確認された．次に，配線幅チェックをネット全体に対して行った．ネットのビア図形数が大幅に減少したため，非常に高速に処理が行えるようになった．今回実験したネットでは，いずれも瞬時に処理が終了している．

さらに 2 点間で枝刈りをして，チェックしたい経路を絞り込めることを可能にした．これにより，従来は人手で取り除いていたテーパリングによる擬似エラーを自動的に除去することができる．

第 4 章　電源配線幅の高速検証システム

4.1　はじめに

LSI のレイアウト設計において, 電源線に十分な配線幅が確保されていないと過剰な電流による溶断などの致命的事故を引き起こすため，配線幅のチェックは極めて重要である．前章までででレイアウトデータからビアを擬似的に削減し，2 点間の配線幅チェックを高速に実行する手法を提案した．そこでは，配線幅検証の対象区間は配線ネット上の開始点と終了点の 2 点で指定される．しかし，この 2 点の座標データは作業者が検証仕様書を見ながら手入力しているため，配線幅チェック自体の高速化は達成できたものの，全体の TAT は改善されないという問題が残っていた．そこで今回，LVS のクロスリファレンス情報を利用し，配線幅チェックの対象となる開始と終了の 2 点の座標を自動的に算出し，その後はバッチ処理にて配線幅チェックを実行するシステムを開発した．これにより人手作業の工程が大幅に短縮され，TAT の改善が期待できる．大規模な実データによる実験の結果，本システムにより 20 箇所の配線幅検証が 30 分ほどで終了し，従来の検証箇所の人手入力による場合と比較して処理時間が大幅に短縮できた．

LSI レイアウト設計ではチップ製造後の回路の動作を保証するために様々な検証が行われている．その中の一つに配線幅の検証がある．特に電源およびグランド配線（以下，PG 配線と略）では，十分な配線幅が確保されていないと配線抵抗の問題や過剰電流により金属配線が溶融して断線や配線短絡に至る致命的な欠陥を引き起こす．このため，PG 配線に対しては特に入念な配線幅検証が必要とされる．しかし PG 配線はチップ全域に渡っており，また，一般に多端子ネットの複雑な構造であるため，配線幅の検証を見落としなく実行するには膨大な処理時間がかかっており，この高速化が最近の超大規模なレイアウトデータを扱う場合の大きな問題となっている．ところで配線幅検証は一般に DRC ツールを用いたレイアウト検証の一工程として行われることが多い．そこで DRC ツールの機能を用いて，PG 配線の検証すべき領域を設計者が指定し，その領域に限定して配線幅検証を行うことで時間短縮を図ることができる．しかし，PG 配線ではテーパリング(Tapering) と称して，配線が分岐するたびに配線面積や電気的性質を考慮して配線幅を調節する処理が行われる．

このため，同一ネットでありながら配線幅が細くなっている部分があるため，領域を対象にした配線幅検証ではこの部分が擬似エラーとして大量に検出されるという問題があった．そこで，指定した領域内の PG 配線上の開始点と終了点の 2 点を設定し，その 2 点間の経路上で主要な PG 配線のみを抽出して検証対象にする手法を提案し，擬似エラーの問題を解決した．さらに最近のレイアウト設計では歩留まり向上のためにダブルビア(Double Via)が多用され，この大量の冗長ビアが配線幅検証での処理時間の悪化を引き起こすことから，検証に無関係な冗長ビアを検証対象のデータから削減する手法を組み入れた．これらの改良により従来手法と比較して約 20 倍の高速化が得られ，配線幅検証そのものに要する時間は大幅に短縮できた．

ところで，配線幅の検証作業では通常，数十箇所あるいはそれ以上の箇所を検査する必要がある．上記の領域指定の方法では，設計作業者が一つずつ領域を指定し，領域内で配線幅を検査する区間の 2 点を定め，実際に検査を実行し，結果を確認するという一連の作業がとられる．これがチップ全体で数十箇所以上に及ぶと，一領域あたりの処理時間は短くてもチップ全体の処理時間は膨大になり，人手に依存する従来手法ではもはや処理不可能となっている．この問題を解決するために，これら全作業をバッチ処理するシステムを開発した．必要な検証対象領域をあらかじめ与えた上で，各領域内での検証の開始点，終了点の 2 点の座標を自動的に求めてバッチ処理で検証を行い，全検証結果をまとめてリスト出力する．これにより VLSI チップ全体の PG 配線幅の検証作業ひいては設計工程の大幅な TAT 向上が期待できる．本システムの特徴は検証対象の等電位ネット指定と検証開始点および終了点の座標決定に LVS の出力のクロスリファレンスファイルを利用することによって全自動化を可能にしたことである．このリファレンスファイルをもとに回路図情報とレイアウト情報とを対応づけることで，従来のレイアウト検証作業のようにレイアウト結果を見ながら作業する手間が不要で，回路図情報から効率よく検証対象を設定でき，検証処理のバッチ実行が可能になった．なお，本論文では説明の都合上，LVS ツールとして文献[32] のツールを前提に述べるが，これに限定されるものではない．

以下，本章では開発したシステムの構成と処理を次節で述べる．4.3 では本システムの実験と評価結果を述べ，最後に 4.4 でまとめを述べる．

4.2 システム構成と検証処理工程

4.2.1 システム概要

本システムの大まかな処理フローを図 4.1 に示す．LVS を実行すると Instance Cross

Reference File (ixf) と Net Cross Reference File(nxf) の 2 つのクロスリファレンスファイルが自動的に作成される．ixf には回路図上のインスタンス（デバイス）名とそのレイアウトデータ上での座標が 1 対 1 の対応で記述されている．この情報から開始点と終了点の座標を取得することができる(詳細は 4.2.4)．

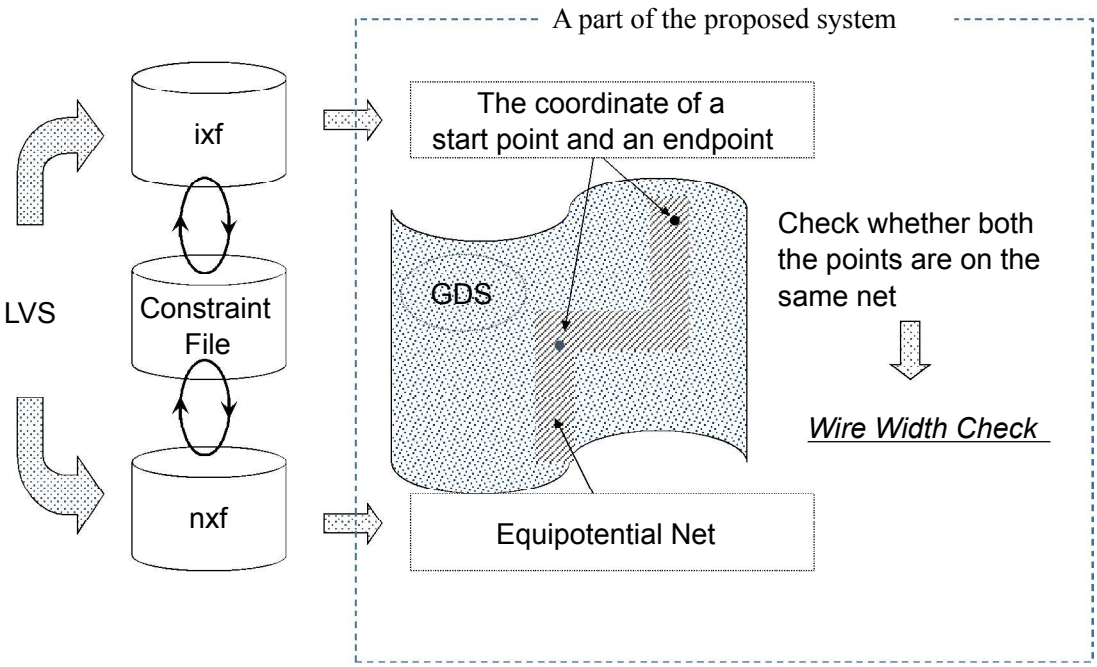
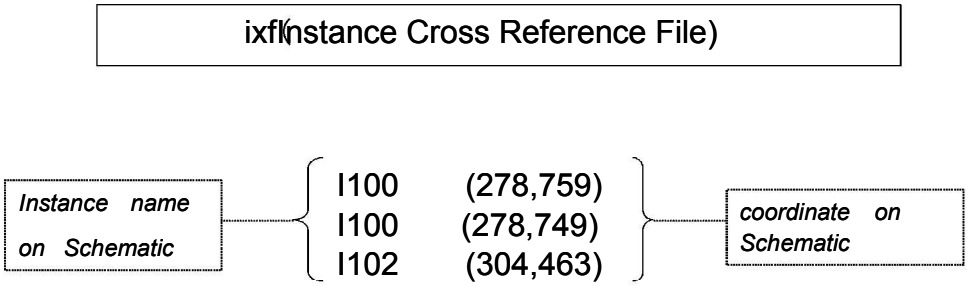
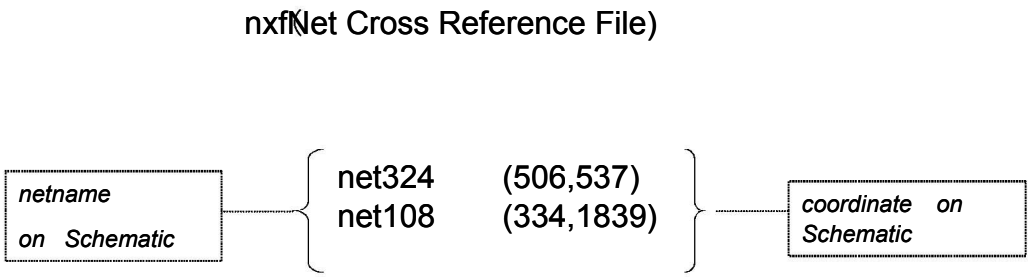


図 4.1 システム概略図

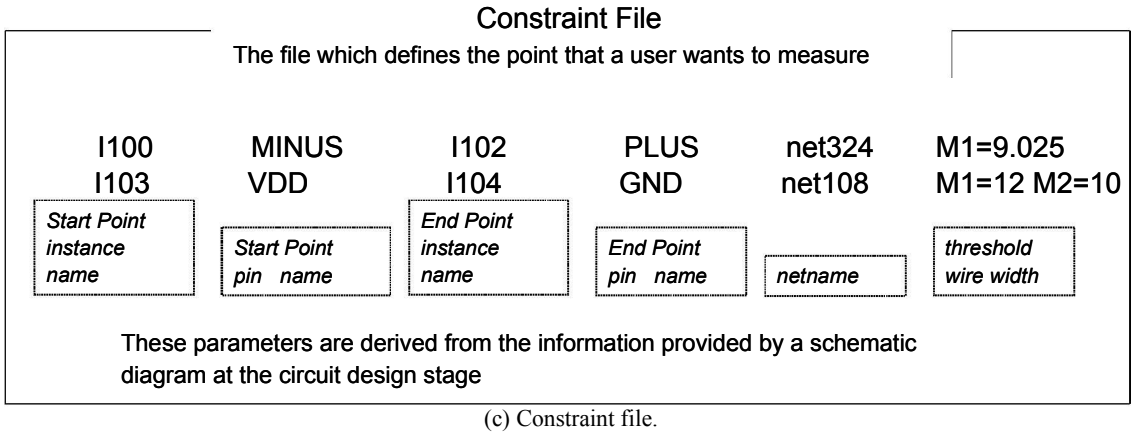
Fig4.1 A system general flow



(a) ixf file.



(b) nxf file.



(c) Constraint file.

図 4.2 入力ファイル書式

Fig 4.2 The format of the input file.

一方nxfには回路図上のネット名とそのレイアウト座標が1対1の対応で記述されている．この座標から等電位追跡を実行し，配線幅検証の対象となるネットを抽出する．ixfおよびnxfには通常，チップ全域のLVS結果の情報が記述されているので，配線幅検証を行う箇所の特定は作業者が制約ファイル(Constraint File)として別途作成する．ixfとnxfから開始点および終了点の座標とネットが抽出されたら，これらの座標がそのネット図形上に含まれているかどうかのチェックを行う．これは同じ名前を持つセルインスタンスが直列に接続しているか並列に接続しているかにより，求めた開始点あるいは終了点がネットに含まれない場合があるためである(詳細は4.2.5)．もし含まれていないことが判明した場合は，再度ixfから同名のセルインスタンスの別の座標を取得し，ネットに含まれる点の座標が得られるまで繰り返す．ネットとそのネット上の開始点および終了点が正しく設定されたら，配線幅検証を順次実行していく．配線幅検証の基準となる閾値は制約ファイルであらかじめ指定される．この閾値を満たさない，すなわち配線幅の違反図形を検出したら出力専用のGDSIIファイルに違反図形を保存する．

4.2.2 入力ファイル

本システムの入力は，LVS出力のixfファイルおよびnxfファイルと，配線幅検証を実施したい箇所を作業者が定義して作成する制約ファイルである．つまりこれら3ファイルは本システム実行前に準備しておく必要がある．制約ファイルには検証対象のネット名，配線幅閾値，検証区間の開始点および終了点のインスタンス名とピン名が記述される．それぞれのファイルの書式を図4.2に示す．なお，制約ファイルに記述する情報は回路図の設計段階で得られるものであり，Virtuoso[40]等の回路図エディタ上で素子のピンを2つ選択し，その情報をテキストファイルに出力することで作成可能である．

4.2.3 等電位ネット抽出

制約ファイルから配線幅検証の対象とするNetnameを取り出し，nxf内のNetnameと照合してネット図形内のある1点の座標を得ることができる．この座標を起点に等電位追跡を実施し，一つの等電位ネットを抽出する．このネットをレイアウトレベルの配線幅検証の対象としてGDSIIファイル[17]に格納する．

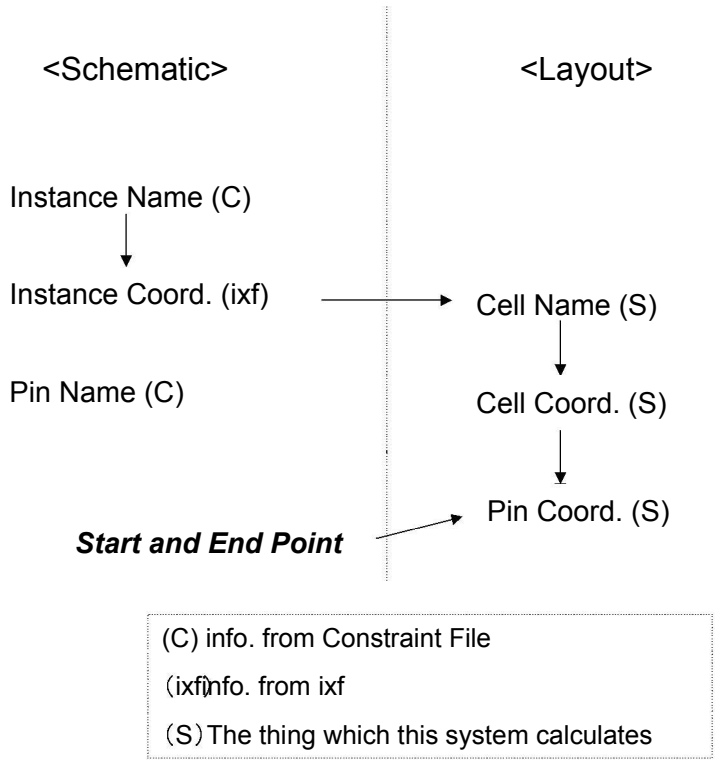


図 4.3　開始点,終了点を求めるまでのフロー（インスタンス,セル,ピンの関係）

Fig 4.3 A flow in search of SP and EP. (Relation of Instance,Cell and Pin.)

4.2.4 4　開始点および終了点の座標抽出

前章でも述べたように，領域を指定した PG 配線の検証処理では大量の擬似エラーが発生する．これを避けるために，開始点と終了点を指定して 2 点間の経路に沿った配線のみ検証を行う．この 2 点の座標は ixf を利用して求められる．図 4.3 に回路図とレイアウトでのインスタンス，セル，ピンの名前とそこから座標を得るまでのフローを示す.

(1)　セルの特定

最初に，制約ファイルに定義されているインスタンス名と ixf 内のインスタンス名とを照合する．このインスタンス名とは回路図上の名前である．図 4.2 のように ixf には同一インスタンス名がレイアウトデータ上に複数配置されている記述が書かれている．先ず，最初の配置場所の座標を取得する．この座標を元に，次にレイアウトデータ上の該当インスタンス座標に置かれているセル名の取得を試みる．ただし，該当座標に置かれているセルは一つとは限らず，通常は階層的に多層のセルが重なっており，上位の階層に位置するセルは下位のセル群を包含する構造で構成される．検証に必要なセルは階層構造の最下位に位

置するセルで，図形やテキストが直接，配置されているセルである．最下位を判断する基準としてセル面積に注目する．すなわち，インスタンス座標に重なっている複数セルのうち，面積が最小のものを採用する．最小面積セルが複数存在している場合は，各セルの重心座標を求め，インスタンス座標からマンハッタン距離の最も近いものにする．重心座標を求めるには，セルの矩形対角頂点座標が必要になるため，次節の方法で求める．

(2) 該当セルのクリップ

次にセルのクリップ（切り出し）を行う．クリップを行う理由は，この後の工程である「セル内をテキスト検索してピンを探す」際に，チップ全体を検索すると多大な時間を要することを回避するためである．クリップのためには，セル位置を特定する座標が必要である．一般にセルの形状は矩形なので，座標を特定するためには，階層構造の最上位セル座標系での対角頂点の 2 点の座標を利用する．GDSII のフォーマットでは，下位セル固有の情報として最上位セル座標系での原点位置と下位セル座標系での原点オフセット位置およびセルの幅と高さが与えられているので，これらを元に特定の座標を求めることができる．ただし，セルの配置は回転(Angle) や反転(Mirror) が施されていることがあるため，これを考慮する必要がある．そこで，図 4.4 のようにセル配置を場合分けし，対角頂点の座標を求める．求める座標は対角頂点の(X1,Y1) および（X3,Y3 ）座標であるが，最上位セル座標系での値でなければならない．セルの原点位置は最上位セル座標でレイアウトデータに記述されている．同様に下位セル座標系での原点のオフセット位置およびセル幅と高さは，セル定義情報（Zone ）から取得でき，図 4.4 中の計算式によって求められる．回転は 90° 単位で行われ，反転と組み合わせて計 8 パターンの配置についてそれぞれの対角頂点座標を求める

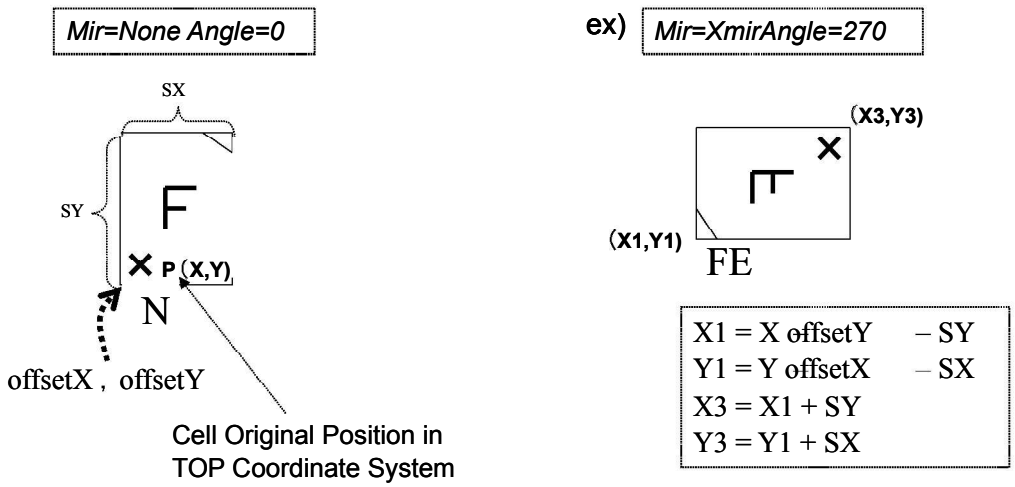


図 4.4 最上位セル座標系でのセル配置座標の計算方法

Fig4.4 Calculation of the cell coordinate.

(3) ピン座標の特定

セルの対角頂点の座標が求まれば，LAVIS を使用して，そのセルだけを別の GDSII ファイル(Binary 形式)にクリップすることができる．次に，その GDSII ファイルを ASCII 形式に変換して,ピン名とピン座標を検索する．ASCII 形式および変換ツールとして本システムでは TOOL 社 Pilot フォーマットと変換ツール lvgds2pilot[41] を使用した．GDSII データ上でのピン名とピン座標は Pilot フォーマットではテキスト記述されている（図 4.5 ）．制約ファイルに記述されているピン名とテキスト記述の文字列が一致すれば，該当ピンが見つかったことになり，テキスト座標を取得する．この座標が開始点または終了点の座標となる．

```
PIL '1.2';
  # This is a sample.
  FILEATTR GDS VER=5 LIBNAME='SAMPLE_TEST.LIB'
 UNIT 1e-09 0.001;
  CELL BLK3 ;
    INSTEXT CPUCELL 50 60 FILE="/disk1/gds/aa.gds" FT=GDS;
    AREA L=13 D=14
      10 60.5
      30 -
      -80.2
      10 -;
  DEFAULT L=33 D=44 F=3 W=2.5 PRE=LB PT=0;
    path 0 0 10 20;
    text "Tool Corp." 10.3 10.8 H=20;
    text meguro-ku20 30.1 PRE=LT;
    text tokyo20 40.5;
  DEFAULT L=22 PRE=CC;
    text "03-5723-8123" 50 60.5;
  ENDCELL;
  ENDPIL;
```

A pin name and the coordinate are described in a text record

図 4.5GDSII データを ASCII 形式で記述した例

Fig4.5An example of GDSII data in ASCII format.

4.2.5 開始点と終了点の対象ネット内包含の判定

(1) 開始点,終了点とネット図形との関係

配線幅検証対象のネット決定(4.2.3) と，開始点および終了点の座標の決定(4.2.4) が終わると，開始点，終了点が当該ネットに含まれているかを確認する必要がある．4.2.4 の(1) で述べたように，現時点で求められた 2 点はあくまでも ixf 内に複数定義されているインスタンスの最初の一つのみを使用して求めたものである．ここで，インスタンスが図 4.6 のような並列接続回路の場合は，ixf で定義された複数のインスタンスのどの座標を採用しても同一の一ネット内に包含される．図 4.6 の例では，ixf にはインスタンス A について左右それぞれ 4 点を共有している．右側の端子でいえば, ixf で最初に現れる (100,400) の座標情報から終了点の座標が決まる．残りの 3 点もすべて同一ネットに包含されるため，配線幅検証を行う区間は異なるが検証自体については問題なく実行可能である．しかし，図 4.7 のようにインスタンスが直列接続の回路の場合は，同じインスタンスでも同一ネットにはならない．そのため，該当ネットに包含される開始点，終了点を ixf 内の他の記述から探し出す必要がある．図 4.7 では，インスタンス A を検索して ixf で最初に現れる座標情報(100,400) から得られる終了点候補は図中の EPx であり，開始点と同一のネットには含まれない．配線幅の検証は配線経路を辿ることであるので，この EPx による終了点座標では配線幅検証の対象区間外となり，配線幅検証自体ができない．なお，図 4.6，4.7 での SP および EP はそれぞれ StartPoint ，EndPoint を意味する．そこで，並列接続か直列接続かにかかわらず，開始点および終了点を定めた後で，それらが当該ネットに含まれるかどうかの判定が必要になる．

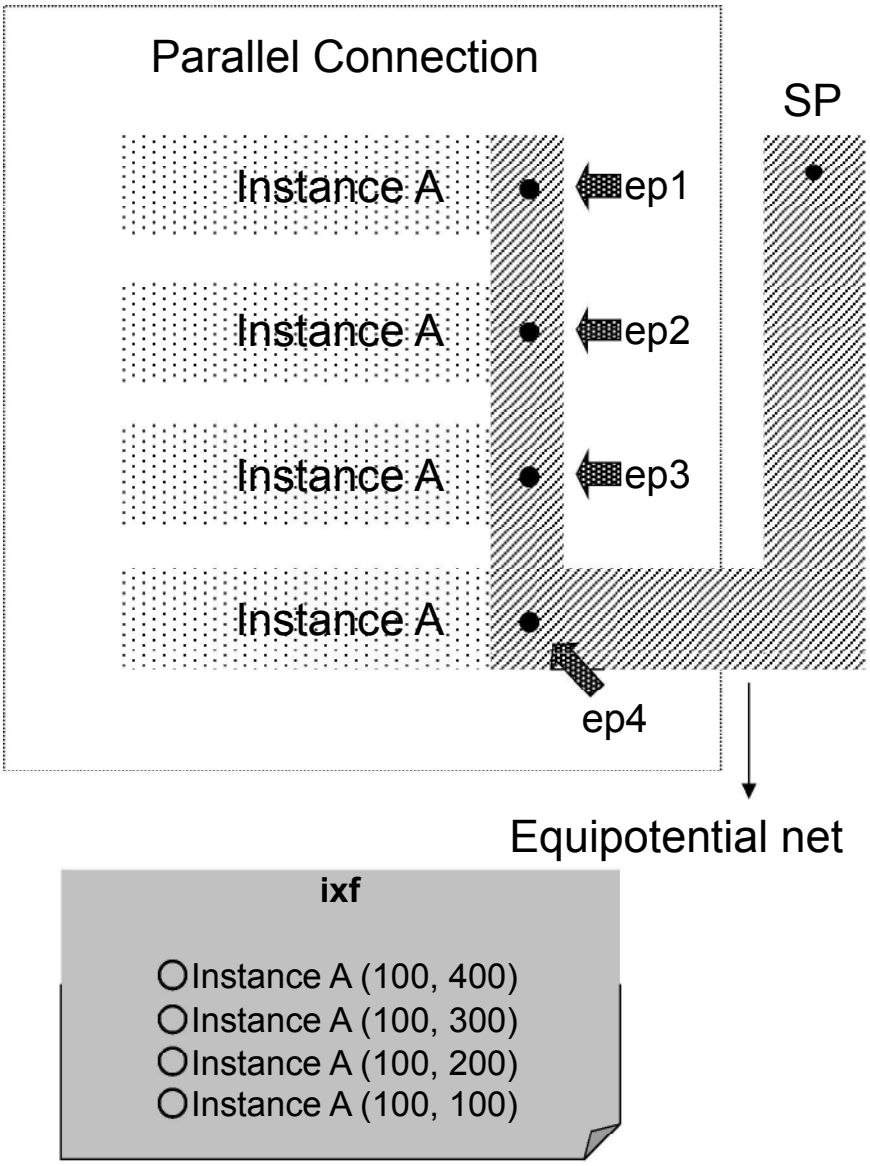


図 4.6 並列接続のインスタンスおよびネット

Fig4.6 Instance and the net of parallel connection.

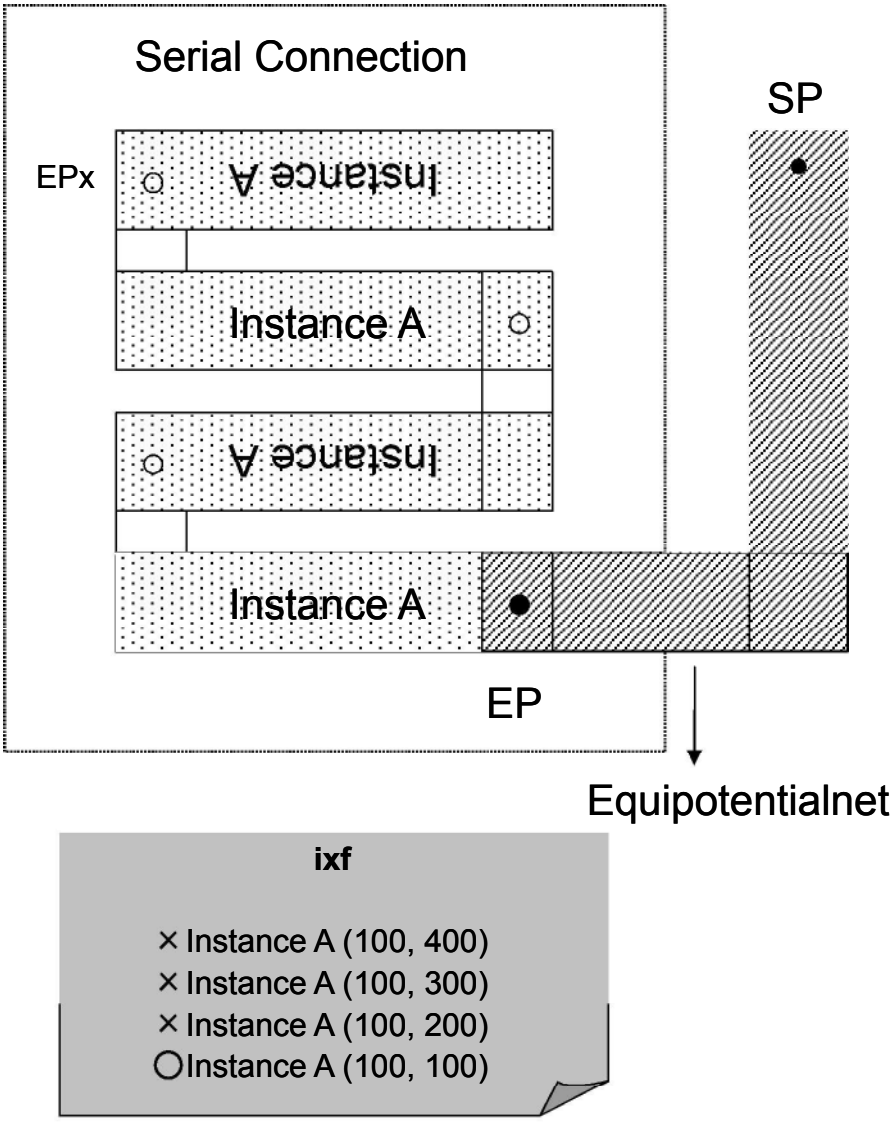


図 4.7 直列接続のインスタンスとネット

Fig 4.7 Instanceand the net of serial connection.

(2)判定手法

ある等電位ネットのレイアウト図形内に特定の座標点が含まれるかどうかの判定は，図4.8 のようにある点 P (X,Y) が直交多角形図形の内側にあるか外側にあるかという問題を解くことと等価である．この問題は「点位置決定問題」として知られており，次のような方法で簡単に判定できる．すなわち，点 P を起点として無限遠点に直線を引き，この直線と図形の輪郭辺との交差回数が奇数回であれば図形の内側，偶数回であれば外側と判定できる．本システムではこの方法を参考にして，まず，点 P が多角形の辺上に重なっていないかを全ての辺に対してチェックし, 重なっていれば, P は多角形の内側とみなして終了する．次に点 P が多角形の辺上にない場合は，P を起点に X 軸平行に補助線を延ばして，多角形の輪郭辺との交差回数を数える．交差回数が奇数ならば多角形の内側，偶数ならば外側とする[42][43] ．ここで，補助線が多角形の頂点と交差した場合には，補助線が多角形の辺と重なっているかどうかのチェックを，X 軸平行で P と同じ高さの辺に対して行い，補助線と輪郭辺が重なっている場合は輪郭辺の始点か終点に接続している辺のもう一方の Y 座標が P より大きいか小さいかを求める. そして大きいものだけをカウントするようにする (図4.9)．あるいは逆に小さいものだけをカウントしてもよい.

以上のようにして，等電位ネット図形内に開始点と終了点がそれぞれ含まれているかを確認する．もし，それらが等電位ネット図形外と判定された場合は，ixf から次の行のインスタンス座標を取得して，4.2.4 の開始点もしくは終了点を求める処理を繰り返す.

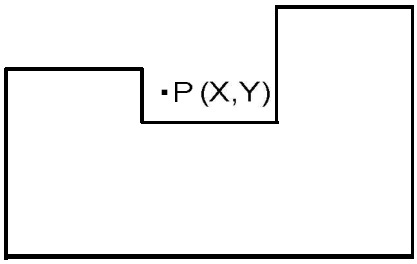


図 4.8 多頂点図形において,任意の点 P の内外判定

Fig 4.8 The inside or outside judgment of an arbitrary point in the polygon figure

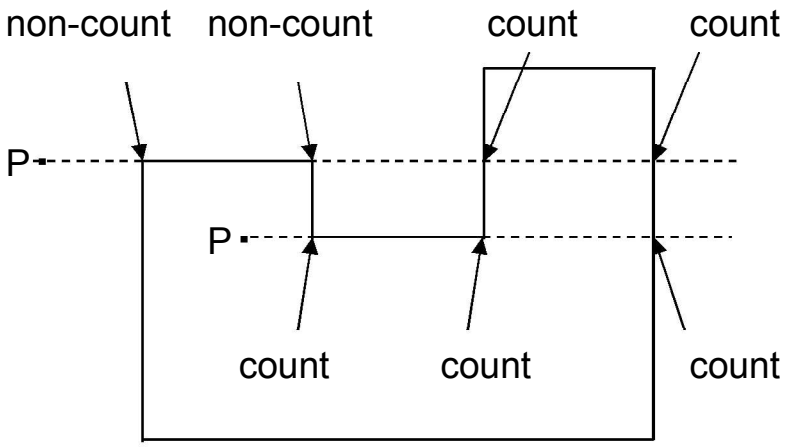


図 4.9 補助線と辺が重なった場合の交差点のカウント方法

Fig 4.9 How to count crossings when an additional line and edge were overlapped.

4.2.6 同一インスタンス定義内でのチェック

等電位ネットとそれに包含される開始点と終了点が求まれば，配線幅検証を実行できる．ところで，図 4.6 のような並列接続の場合は ixf に記述されているどのインスタンス座標が選ばれても，求められた終了点は同一の等電位ネットに含まれるため，配線幅チェックは実行可能であると 4.2.5 の(1)で述べた．しかし，2 点間で配線幅検証を実行する場合，図 4.6 の ep1 点を採用するのと ep4 点を採用するのとでは，対象区間が異なる．ep4 点が採用された場合 ep1-ep4 間は配線幅検証の対象から外された状況になる．そこで，ixf 内に同名インスタンスの記述が複数存在した場合は，開始-終了点間のチェックだけではなく，同じインスタンスグループの中でのチェックも行う必要がある．図 4.6 の場合は 4 つの InstanceA からピン位置の該当座標を算出し, ep1 を起点として, ep1-ep2 間, ep1-ep3 間, ep1-ep4 間という具合に, ixf に記述された(インスタンス数-1)回のチェックを行う．実際，図 4.6 の場合であれば，ep1-ep4 間のチェックを 1 回実行すれば十分である．しかし，ixf に記述されている順番は必ずしもネット上での座標の昇順，降順ではなく，さらにピンの座標位置からでは，ネットの形状も考慮したうえで離れている 2 点は特定できない．そこで冗長ではあるが，この方法を採用している．

4.3 実験と考察

表 3.1 に示した 1.7GB の GDSII データに対して，本システムを使用して配線幅検証を実行した．実験に用いた回路規模は，レイアウトデータに換算して ixf に約 10 万インスタンス，nxf に約 4 万ネットの記述を含むものである．4.2.1 で述べたように，配線幅検証の閾値は制約ファイルで指定されるため，この閾値を変更することにより，配線幅違反の状態を作成することができる．M1 層の配線幅閾値を変更して実験した結果，M1 層の配線幅閾値を実際のレイアウトデータの配線幅より小さく設定した場合には，エラーは検出されなかった．対して，配線幅閾値を大きく設定した場合はエラーとなり，M1 層配線部分の図形が別 GDSII として出力された．また，制約ファイルに 20 箇所の検証箇所を指定したところ，20 箇所のネットとそれぞれの開始・終了点を抽出するまで約 2 分，配線幅の検証を約 27 分で終了した．

一方，同様の検証作業を本システムを用いずに人手で行った場合，作業者はまず，回路図との突合せをしながらレイアウトデータをテキスト検索し，開始点と終了点に相当する位置を求める必要がある．また， DRC ツールが出力する数千箇所のエラーから擬似エラーを除外しながら目視で正味のエラー箇所を探す必要があった．これらの作業には数日が必要であったが本システムを導入することにより，数十分で終了することから，本システムの有効性が確認できた．

4.4 結論

LVS の出力であるクロスリファレンスファイルを活用して，大規模な VLSI レイアウトデータの配線幅検証を効率よく実行するシステムを開発した．特に，従来は検証作業の前準備のために入力データ作成や操作をマニュアルで行っていたが，本システムでは LVS 出力を利用することで配線幅検証箇所の開始点，終了点を自動的に求めることが可能となり，人手介入を極力なくすことに成功した．また，配線幅検証の工程そのもののスピードアップのために，開始点と終了点を結ぶパスから外れた部分の枝刈り操作と，複数ビアによる等電位ネット構成から冗長なデータを排する処理を導入した．これらの工夫により，従来の人手による検証作業に比べて大幅な時間短縮を得た．実データを用いた実験では，従来の人手を介して行う方法では数日かかっていた検証時間が数分に短縮することができた．

第 5 章　DRC検証作業の支援システム

5.1　はじめに

LSI のレイアウト検証の一つである DRC 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[44]．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 ツールと呼ばれる EDA 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 フォーマット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．その際は下位階層で発生しているエラーも TOP セルから見た状態でバッチ的に画像保存できる．このように下位階層のエラー図形を上位階層画像に重ねて自動保存できる EDA ツールは，現在まで他に存在しない．

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

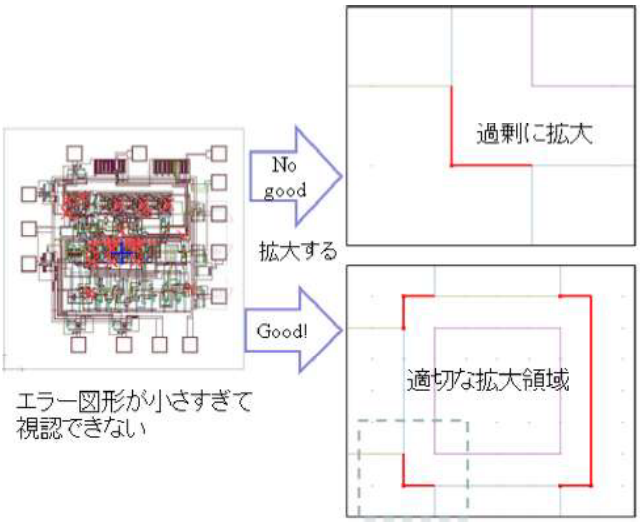


図 5.1　FIT 画面でエラー図形を重ねている表示（左），一つの図形を拡大した表示（右上），適切なグルーピングを行った表示（右下）

Fig5.1 Display (left) of error figure overlaid on the FIT display, one figure displayed in enlarged state (top right), display after appropriate grouping (bottom right).

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 5.1 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．

そこで最初に階層的クラスタリング法を使用して，エラー図形を“適度なまとまり”ごとに画像保存を行う方法を提案した．その結果，所望の画像を得ることは可能になったが，エラー図形の増加に伴い，階層的クラスタリングに要する処理時間が問題になってきた．そこで次に，最初のシステムで使用していた GROUP-SIZE　の値を利用してクラスタリングアルゴリズムの高速化を図った．GROUP-SIZE　とは，複数のエラー図形を一つのまとまりとするための領域サイズを表すパラメータである．これを事前に与えることにより，n をエラー図形の総数としたとき階層的クラスタリング法では $O(n^3)$であった時間計算量を，次に提案する高速化手法ではほぼ線形にすることが可能になった．また，出力結果に関しても

視認しやすい“まとまり”が得られていた．どちらの方法においても，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 Intellectual Property （IP) 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．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．

以下，5.2 で本システムに概要について述べ，5.3 では本システム処理内容の詳細について述べる．5.4 では階層的クラスタリング法について述べ，5.5 では階層的クラスタリング法の問題点と単リンク拡張法について述べる．5.6 では２つのアルゴリズムを用いた実験結果を評価し，5.7 でまとめる．

5.2 システム概要

5.2.1 使用ツール

DRC ツールは[39] を使用した．本システム開発には前章までと異なり TOOL 社 LAVIS-plus[19] 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[45] 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 を介して Python 言語[22] にてプログラミングを行った．

5.2.2 処理フロー

本システムの大まかな処理フローを図 5.2 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

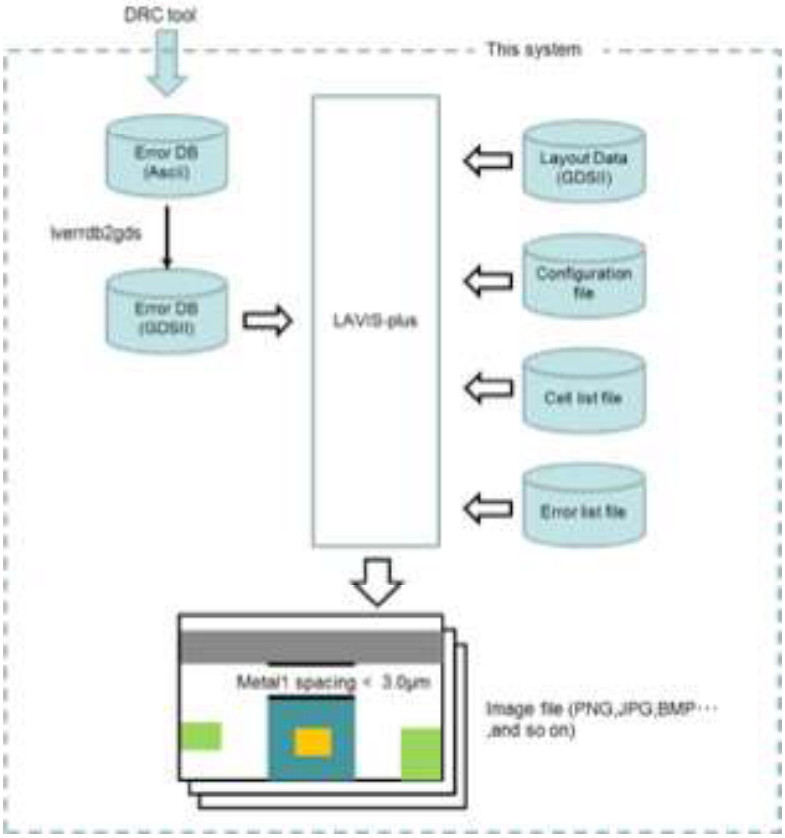


図 5.2　システム概略図

Fig5.2　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            10
(6)ERRORFLG-COLOR           Red
(7)ERRORFLG-WIDTH           1
(8)OUTPUTFILE-FORMAT        PNG
(9)GROUP-SIZE               20
(10)IMAGE-SIZE              500x500
(11)OUTPUTDIR               /home/xxx/output
(12)ALGORITHM               HIERARCHICAL
```

図 5.3　設定ファイル書式

Fig5.3　Format of the configuration file.

を画像ファイル形式（PNG ，JPEG ，BMP 等）にして出力する．設定ファイルとして図 5.3 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．

(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 5.4）.ここではそのファイル名を指定する．バッチ処理の過程で，この ASCII 形式のファイルは GDSII 形式に変換される．詳細は 5.3.1 に記す．

```
TEST6 1000
R1.5
4 4 3 Nov  2 16:42:07 2014
Rule File Pathname: drc.rul
Nwellenclosure of Ndiff< 0.5um
(Substrate/Well contact Active to Nwelledge)
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 5.4 ASCII エラーDatabase 書式

Fig5.4 Format of the ASCII Error-DB file.

(3) CELL-LIST-FILE は図 5.5 左のようにセル名を 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．詳細は 5.3.1 に記す．

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

```
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 5.5　セルリストファイル書式（左），エラーリストファイル書式（右）

Fig5.5　Format of the cell list file(left).Format of the error list file(right).

(4)　ERROR-LIST-FILE　は図 5.5 右のように，画像に出力するエラールール名とその表示色を指定する．

(5)　WINDOW-MARGIN　は画像保存する際, 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　はクラスタ間の併合距離を指定する．詳しくは 5.4.2 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．

(12)　ALGORITHM　は階層的クラスタリング法か単リンク拡張法のどちらを適用するかを指定する．詳しくは 5.6 に記す．

5.3　システム内部処理

5.3.1　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．この GDSII ファイルは図 5.6 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 5.4 内のエラー内容の文字列（下線部）も GDSII の TEXT[17] 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．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．

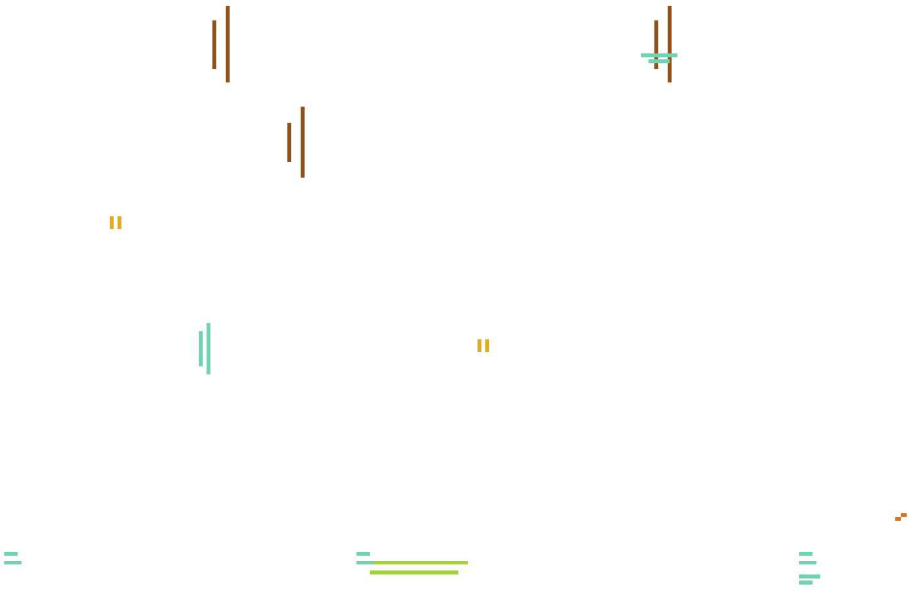


図 5.6　DRC から出力されたエラー図形（GDSII)

Fig5.6　Output error figures from DRC.

(a)　　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 ファイルを該当セルに重ね合わせる．

(b)　　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．

5.3.2 TOP セルからの俯瞰画像の自動保存

5.3.1 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．5.3.1 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 5.7 の左上の Cell B のようなイメージになっている．

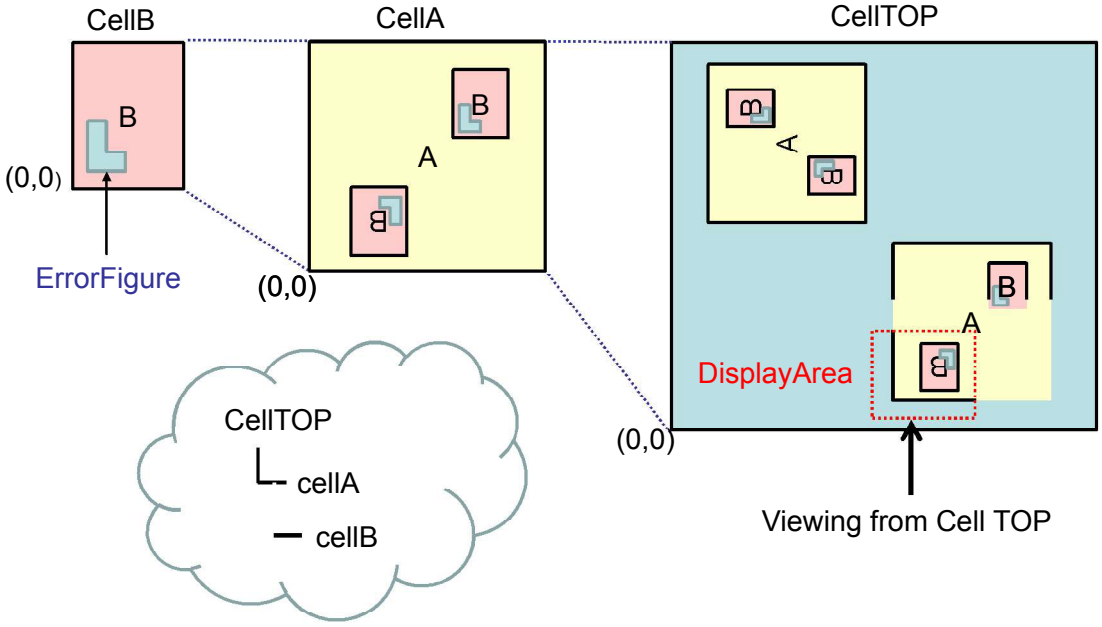


図 5.7　セル階層と画像保存領域

Fig5.7　Cell tree and display area.

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 5.7 右の Cell TOP 内の Display Area と定義されている箇所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．実際に作成した参考画像として図 5.8 および図 5.9 を示す．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．作業者が操作することにより，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セル単位で重ね合わせることは LAVIS[18][46] など従来の EDA ツールでも可能である．しかし，下位セルに重ね合わせられたエラー図形を，TOP セルから俯瞰した画像でバッチ的に自動保存することはできなかった．

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．そこで 5.4. の手法によりエラー図形をグループ化して画像保存を行う．

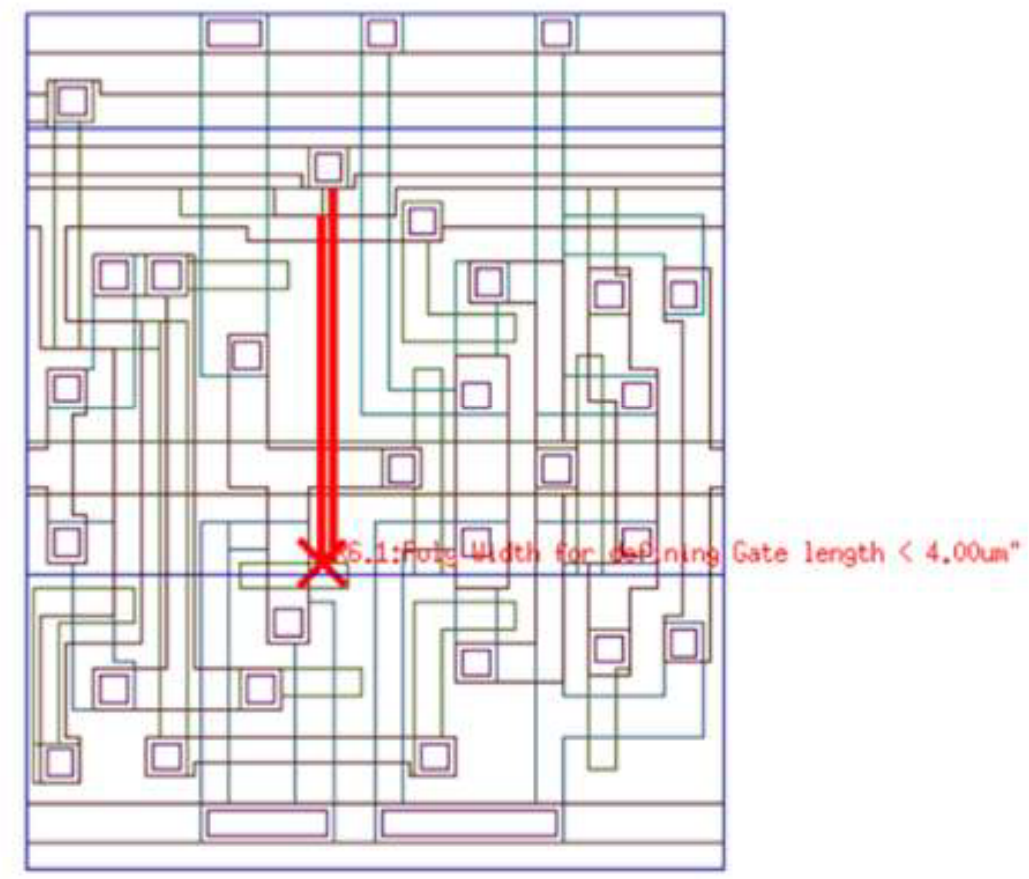


図 5.8 エラー図形とレイアウトデータ上の特定セルとの重ね合わせ（LOCAL セル指定）

Fig5.8Overlay of error figures and layout data (specified LOCAL cell).

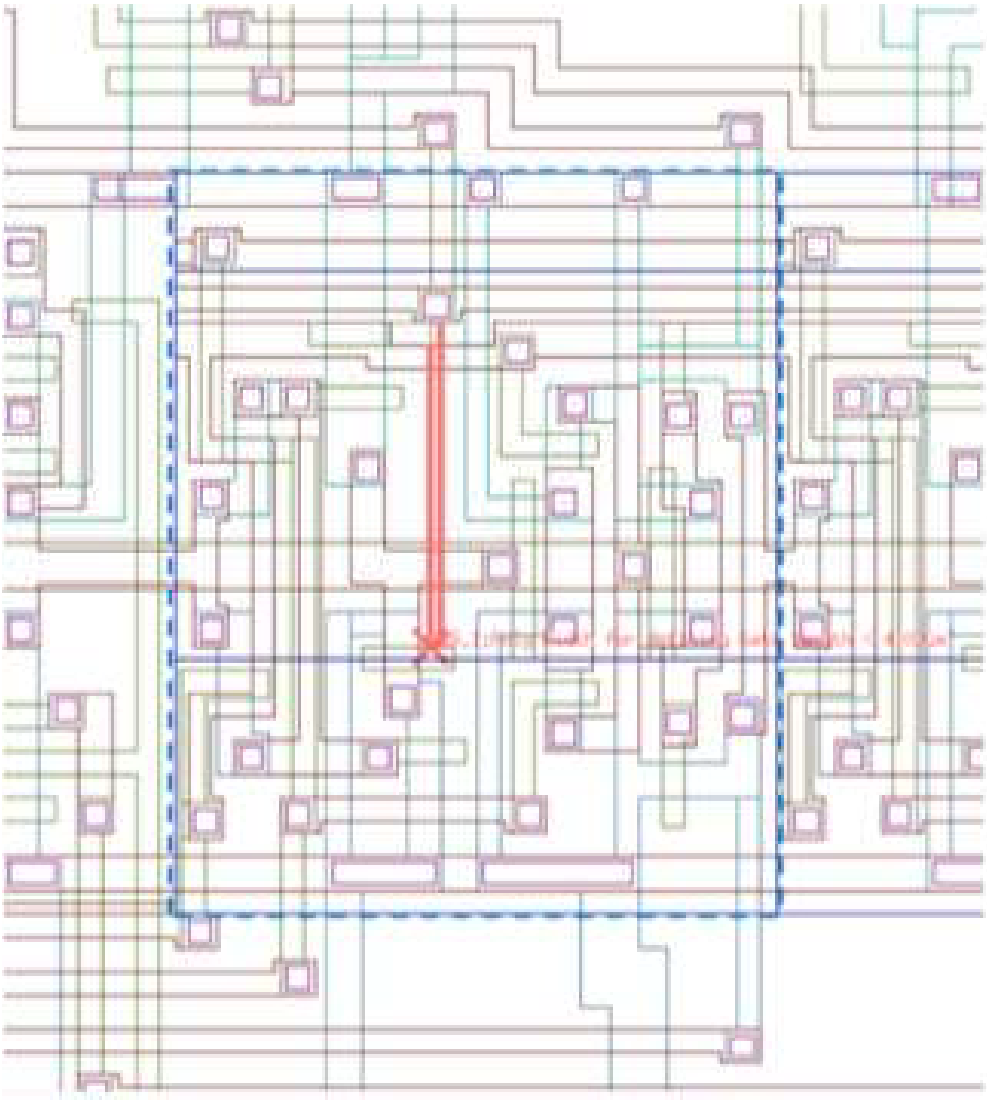


図 5.9 エラー図形とレイアウトデータの重ね合わせ（TOP セル指定，点線部が一つのセル）

Fig5.9Overlay of error figures and layout data (specified TOP cell).

5.4　グループ化のアルゴリズム

5.4.1.1　階層的クラスタリング法のアルゴリズム

当初，本システムのエラー図形のグループ化のために統計学などで広く使われている階層的クラスタリング法を採用した．階層的クラスタリング法は個体を一組ずつ併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[47]．これは重み付きグラフの最小全域木を求める方法と類似している．先ず，5.3.1 で変換した GDSII のエラー図形データ（図 5.6）から重み付き完全グラフを作成する．グラフの頂点はエラー図形の座標から求める．DRC ツールが出力するエラー図形は図 5.10 のような edge か polygon の二種類である[39]．edge ならば中点，polygon ならば外接矩形中点をグラフの頂点の座標とする（図 5.10 では×で示した箇所）．一方，グラフの辺の重み $w(e)$ は頂点間のユークリッド距離とする．図 5.11 に 6 個のエラー図形を頂点とした重み付き完全グラフ $K_n=(\{1,2,\cdots,6\},)E$ を用いたクラスタリングの例を示す．

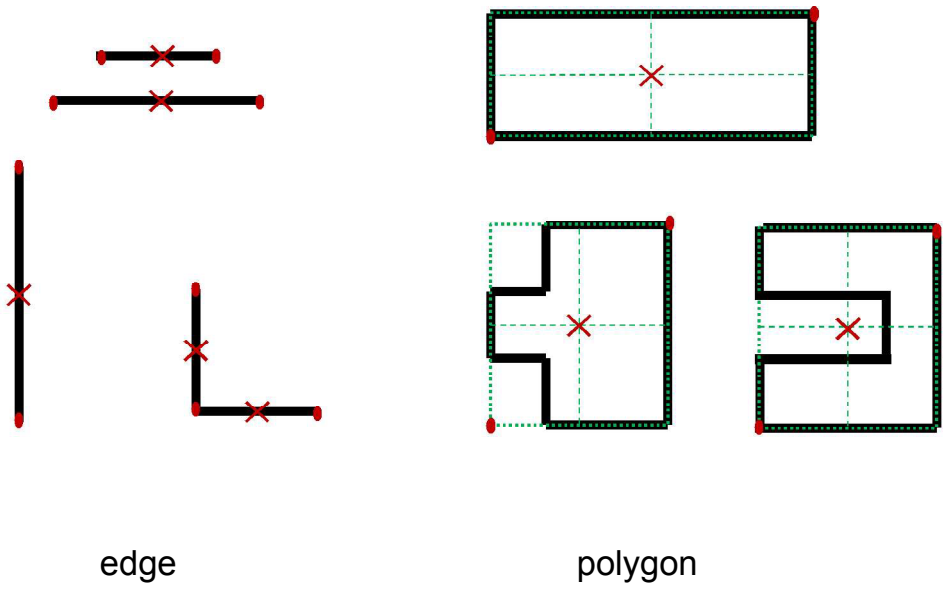


図 5.10　DRC ツールから出力されるエラー図形種別

Fig5.10　Error figure types that are output from DRC tool.

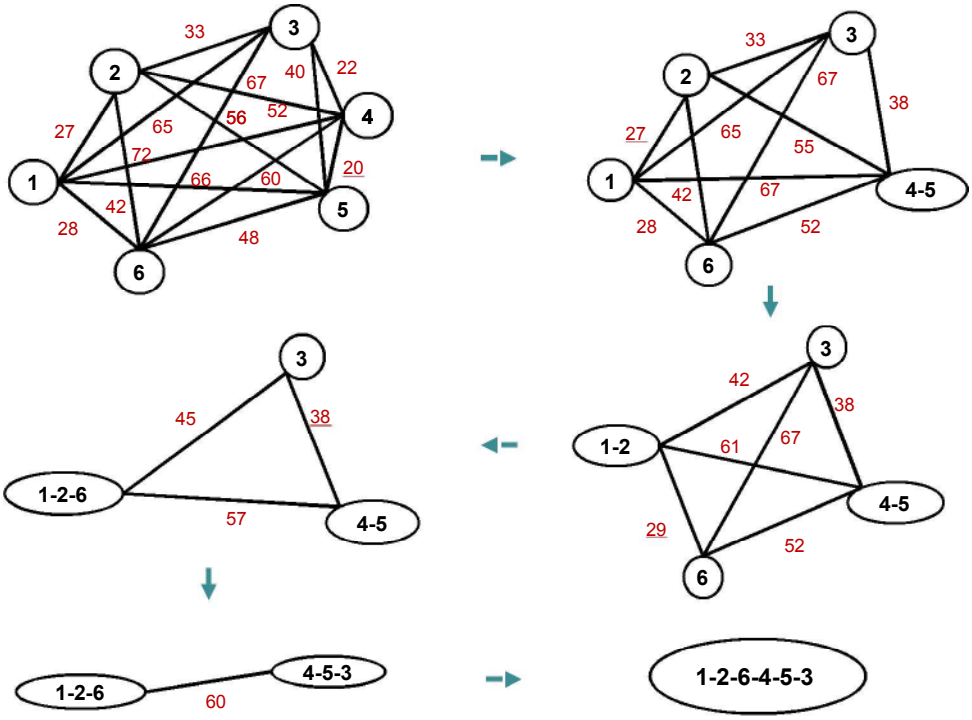


図 5.11　階層的クラスタリング法

Fig5.11　Hierarchical clustering method.

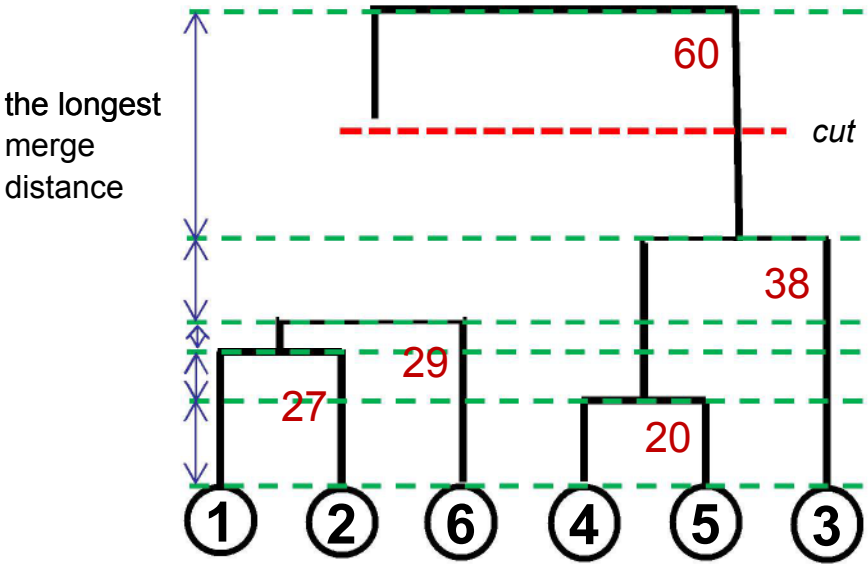


図 5.12　得られたデンドログラム

Fig5.12　Obtained dendrogram.

このグラフ Kn を以下の手順でクラスタリングする．グラフ Kn の最も重みの小さい辺 e の端点を併合する．併合した新たな頂点は併合前の頂点の外接矩形中点になる．新しく発生した頂点に対して辺の重みを再計算してグラフを更新し，頂点が 1 個になるまでこれを繰り返す．この過程は図 5.12 のようなデンドログラムで表すことができる．デンドログラムとはクラスタ生成プロセスを木構造で表現したものである[48]．デンドログラムの縦軸は辺 e の端点を併合したときの重み $W(e)$ になる．（以下，併合距離と呼ぶ）ここで枝をカットすると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．本システムでは最も長い枝をカットすることとし，図 5.12 では最終的に２つのクラスタに分割している．階層的クラスタリング法では，作業者が分割数や併合距離といったクラスタに関するパラメータを一切与えることなく処理が完結できることが特長である．一方で，併合距離を作業者が指定できるモードも作成した．これが 5.2.2(9) で述べた GROUP-SIZE を使用する方法である．GROUP-SIZE を指定することにより，併合距離が GROUP-SIZE を上回ったらクラスタリング処理を中断し，その時点でのクラスタを出力することにした．その場合はデンドログラムを生成する必要はなく，頂点が 1 個になるまでクラスタリングを行う必要もないので，その分処理が高速になる．

図 5.13 および図 5.14 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 5.10 で保持していた 2 頂点から中点×を求めて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 5.13 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 5.14 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．このように本システムではクラスタの中点を求め，クラスタ間距離の算出に使用した．一般的に階層クラスタリング法では，この他にも最短距離法，最長距離法，重心法，重み付き平均法など多くのクラスタ間の距離計測方法が存在する[47][48]．本システムでクラスタ中点間の距離を計測する方法を採用した理由は，クラスタの対角頂点 2 点の座標のみを保持していればよいことと，他の方法と比較して計算量が少ないことによる．

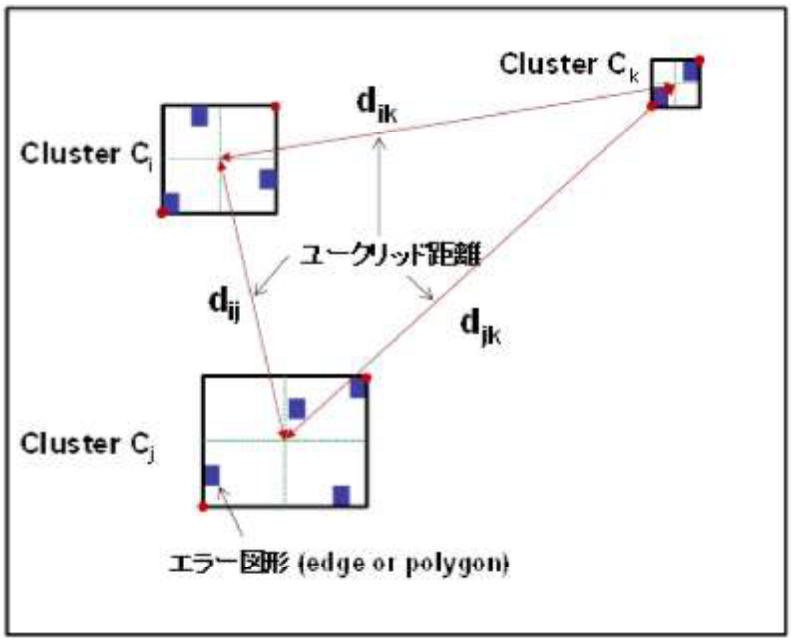


図 5.13 階層クラスタリング法　実行前

Fig5.13Before mergingthe clusters.

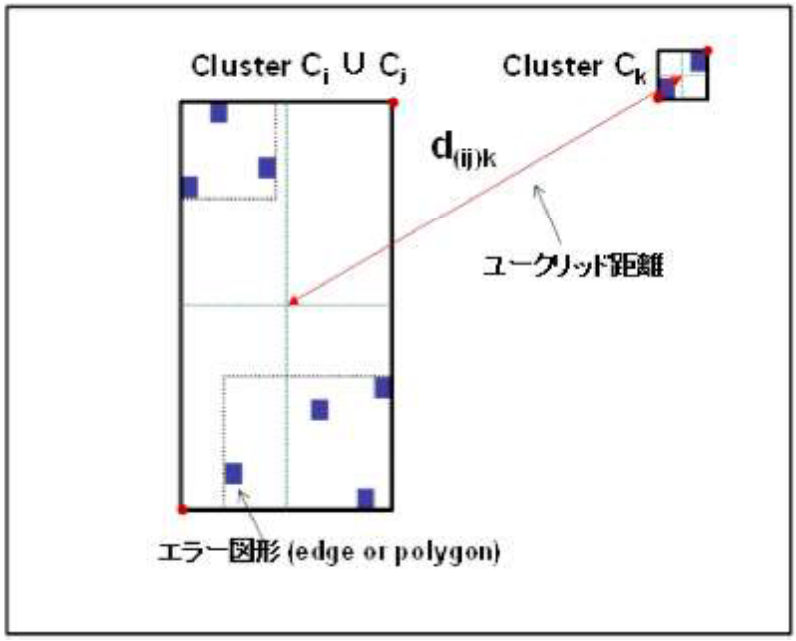
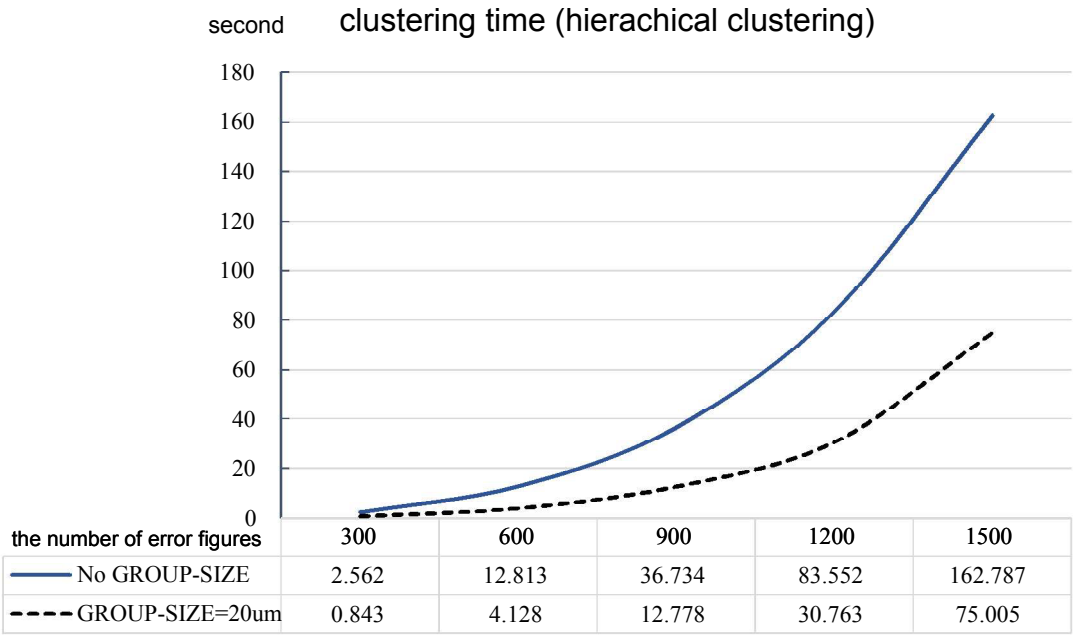


図 5.14.階層クラスタリング法　実行後

Fig5.14Aftermergingthe clusters



the number of error figures	300	600	900	1200	1500
No GROUP-SIZE	2.562	12.813	36.734	83.552	162.787
GROUP-SIZE=20um	0.843	4.128	12.778	30.763	75.005

図 5.15　階層的クラスタリング法の処理時間

Fig5.15　Processing time by hierarchical clustering method.

表 5.1　実験に使用したレイアウトデータ

Table5.1Experimental layout data.

ファイルフォーマット	GDSII
ファイルサイズ	1.78(MB)
ファイルユニット	1e-9(m)
総レイヤ数	33
チップサイズ	1280(um) x 1290(um)

5.4.2　階層的クラスタリング法の処理時間と問題点

5.4.1 に述べた階層的クラスタリング法の処理時間を計測した．使用したレイアウトデータの情報を表 5.1 に示す．このレイアウトデータに対し{@ Poly overhang of Active < 3.0 ENC ACTIVE POLY < 3.0 ABUT < 90 SINGULAR}　という条件[39] で DRC を実行した．拡散層と直行している Poly 層のはみ出しの長さが 3um 未満の箇所をエラーとするという

意味になる．この結果出力されたAscii エラーDB ファイルのエラー図形を元に，乱数でチップ全体に300~1500 までエラー図形を増加させたエラーDB ファイルを作成した．その際5.4.4 で述べる2 図形のペアは崩れないように留意した．実運用ではエラー図形数が1500を超えるケースも想定されるが，エラー図形数と処理時間の時間計算量を調べる目的では適切な個数と考えた．上記のデータに対し，5.4.1 のアルゴリズムによりクラスタリングを実行した場合の図形数と処理時間を図5.15 に示す. 処理時間には図5.10 に記したエラー図形の中点を求める処理も含まれている．実線グラフはGROUP-SIZE を指定せずに，クラスタが1 になるまで実行した場合の時間である．一方，点線グラフはGROUP-SIZE を20um として，クラスタ併合距離が 20um を超えた時点で処理を終了した場合の時間になる．どちらも図形数の増加に伴い，急激に処理時間が増大する．これは完全グラフの辺の数$|E|$×頂点数$|V|$-1 の演算が発生しているためである．エラー図形数をnとしたとき

完全グラフの辺の数$|E| = (n-1)n/2$,

完全グラフの頂点の数$|V| = n$,

となり時間計算量は$O(n^3)$と見積もれる[49]．階層的クラスタリング法のデータ構造やアルゴリズムを改良した場合でも，$O(n^2 \log n)$を要している[50]．LSI 設計の初期段階ではDRCから出力されるエラー図形数は数万以上に及ぶケースもあり，線形時間で処理できることが大きな課題であった.

5.4.3 GROUP-SIZE を利用した単リンク拡張法

階層的クラスタリング法では処理を途中で中断するために GROUP-SIZE を設定できるようにしていた．ところで階層的クラスタリング法は観測データ以外の情報を外部から与えずともクラスタリングを行えることが，その最大の長所である．さらに，分布の不規則な多数のデータに対して，作業者がクラスタ間の適切な併合距離，つまり GROUP-SIZE をあらかじめ指定できるかという疑問があった．DRC 検証作業者を対象に GROUP-SIZE に関して調査を行った結果，チップサイズとプロセス（MOS トランジスタ最小ゲート長）があらかじめ分かっているので，適切な GROUP-SIZE を指定することは可能という結論を得た．そこで，事前に指定した GROUP-SIZE の値を利用して線形時間で効率よくクラスタリングを行う以下のアルゴリズムを提案する．（単リンク拡張法と呼ぶ）

(前処理) 5.4.1 および図 5.10 に記したエラー図形の中点を求める処理を行う．したがって，以降の処理は一つのエラー図形に対し，1 点で表現した座標で実行する．

(Step.1) エラー図形を X 座標でソートする．

(Step.2) 基点となる未処理の任意のエラー図形を取り出す．

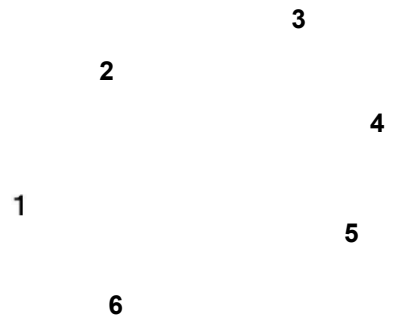
(Step.3) 基点を中心として一辺が GROUP-SIZE の 2 倍の長さの正方形（BOX 領域）を作成する．

(Step.4) Step.1 の結果を利用し，BOX 領域内に X 座標が含まれているエラー図形を取り出し，その中から Y 座標が領域に含まれているものをさらに取り出す．ここで新たなエラー図形が存在すれば，そのエラー図形を基点と同グループにする．

(Step.5) Step.4 で新たなエラー図形が取り出された場合はその点を基点として Step.3 へ．取り出されたエラー図形がなければ現在処理中のグループを確定して Step.2 へ．その際，グループに確定された点は以降の探索候補から外す．すべてのエラー図形を処理したら終了する．

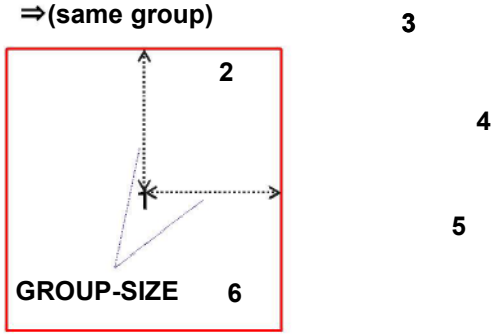
図 5.16 の例では平面空間上に 6 個の点があり，これらを上記の(Step.1) から(Step.5) を用いてクラスタに分割する過程を示している．

(1)Sort the vertex by x-coordinate



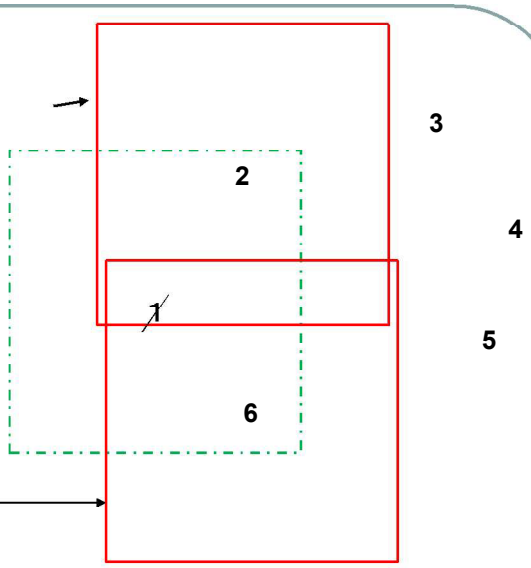
X: 1 < 2 < 6 < 3 < 5 < 4

(2) Create a box with the center at No.1.
⇒ 2and6are included the box.
⇒(same group)



GROUP-1:{1,2,6}

(3)Create a box with the center at No.2.
⇒There is no vertex in the box
except for 1 already grouped.



(4)Create a box with the center at No.6.
⇒There is no vertex in the box
except for 1 already grouped .

The group {1,2,6} is determined at this time.

GROUP-1:{1,2,6}

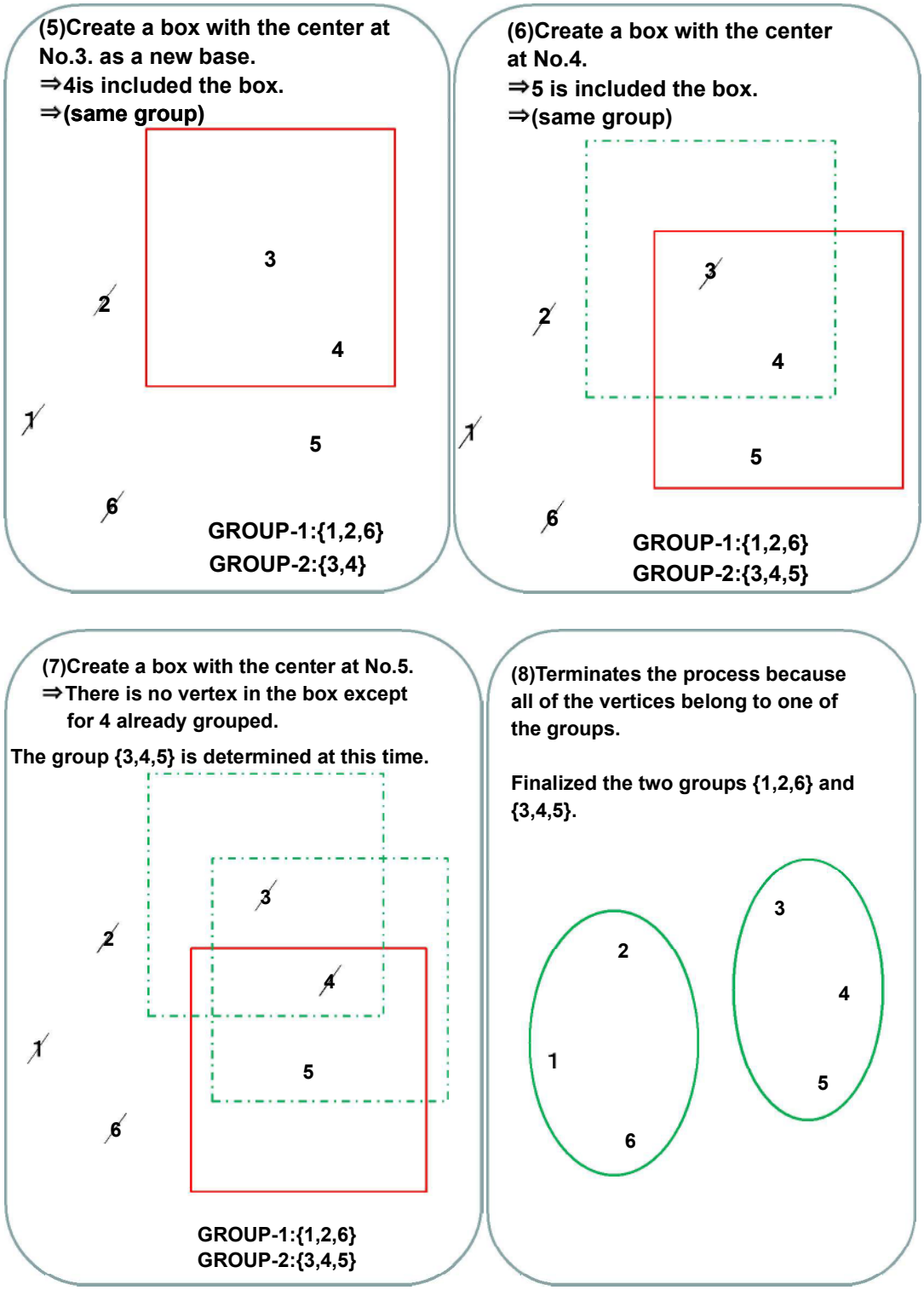


図 5.16　単リンク拡張法

Fig5.16　single link extension method

一方，時間計算量に関して言えば，エラー図形の中点を求める前処理は $O(n)$．(Step.1) は TimSort [51] を利用し，最良ケースで $O(n)$，最悪ケースで $O(n\log n)$になる．(Step.2) ，(Step.3) は $O(1)$，(Step.4) は GROUP-SIZE を極端に大きく設定し，BOX 領域内に全ての点が含まれるケースで $O(n)$となるが，その場合はその時点で処理終了する．また通常，一度クラスタリングされたエラー図形は以降の探索候補から外されるが，GROUP-SIZE を極端に小さく設定し，1 クラスタに１つのエラー図形しか入らない場合は，1 回の探索につき 1 個しか候補から減らない．したがってその場合の合計比較回数は $n(n-1)/2$ 回となり(Step.4) の時間計算量は $O(n^2)$ となる．また(Step.4) に Range tree[52] を適用した場合は GROUP-SIZE にかかわらず(Step.4) の計算量は $O(\log n)$と見込まれる．しかし，(前処理) から(Step.5) までを通して考えると，(前処理)の係数が大きく(Step.1) 以降の処理の係数は小さいことから，全体でほぼ線形時間に近くなると見込まれる．

5.4.4 4　他手法の検討

GROUP-SIZE　が与えられていてクラスタリングを行う場合，通常は平面空間を GROUP-SIZE　でメッシュ状に分割し，その領域単位でクラスタ分割する方法が考えられる．しかし，DRC では配線幅や配線間隔の違反は 2 図形がペアとなって出力され，エラーを指示する場合が多い（図 5.17 ）．だが，そのペアという情報は 5.3.1 で ASCII エラーDB ファイルを GDSII に変換する過程で失われてしまう．ペア図形は距離的に近い場所に存在するのが普通だがメッシュ状に機械的に分割した場合，ペア図形も分断してしまう可能性がある．その状態でエラー画像が取得された場合，エラー箇所の特定が困難になるためメッシュ分割による手法は採用できない．

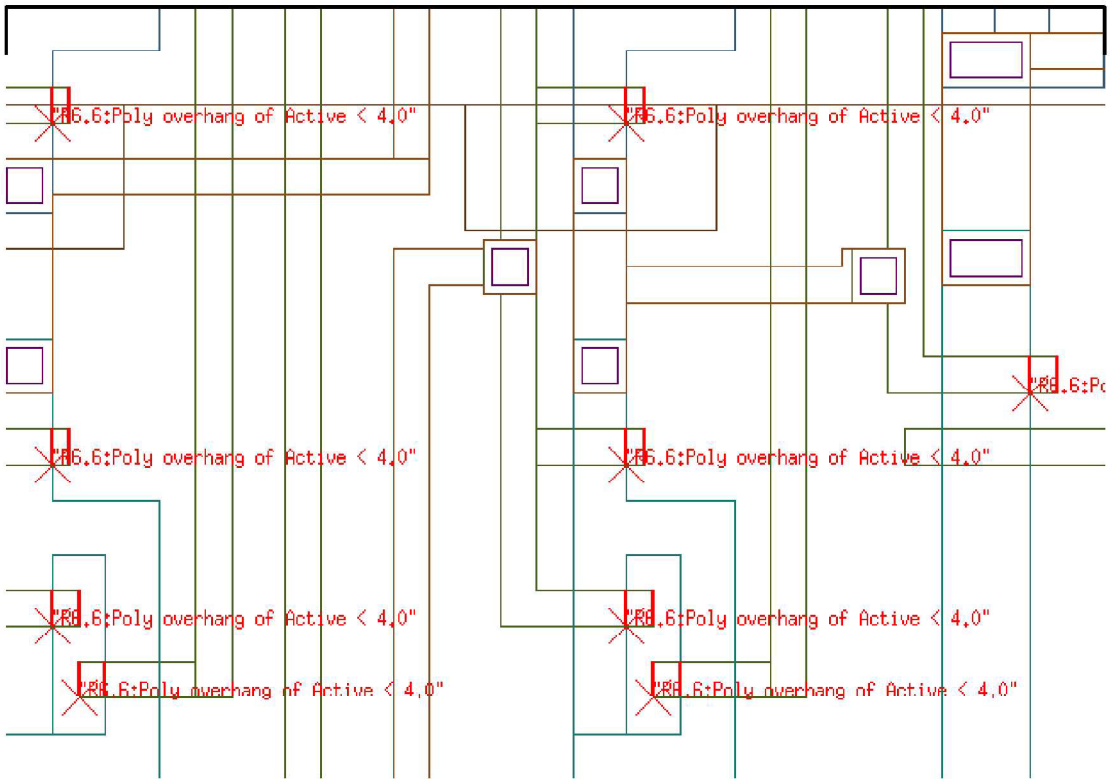


図 5.17　2 図形のペアでエラーを表す例

Fig5.17　Example that represents the error in two figures of pair.

5.445 5　階層クラスタリング手法と単リンク拡張法の選択

階層クラスタリング手法と単リンク拡張法は全く同じクラスタリング結果が得られるわけではない. 例えば図 5.18 および図 5.19 のように等間隔でエラー図形が並んでいるような場合，GROUP-SIZE　を指定しない従来手法では{1,2}と{3,4,5}の二つのクラスタに分割される（図 5.18 ）.一方，d＜GROUP_SIZE＜2d　のときの単リンク拡張法ではすべてが一つにまとめられる（図 5.19 ）．したがって単リンク拡張法では，チップの全面に渡って GROUP-SIZE　以内でエラー図形が検出された場合の最終的な出力画像は，クラスタ分割されずに FIT 画面となる．また，階層クラスタリング法には GROUP-SIZE　の指定が必須でないという利点がある．したがって，エラー図形が少ない場合は単リンク拡張法を実行せずに階層的クラスタリング法を選択したい場面も発生する．そこで，本システムでは作業者は検証状況により，どちらのアルゴリズムを使用するか選択可能にした．図 5.3(12) の ALGORITHM　にて指定を行う．

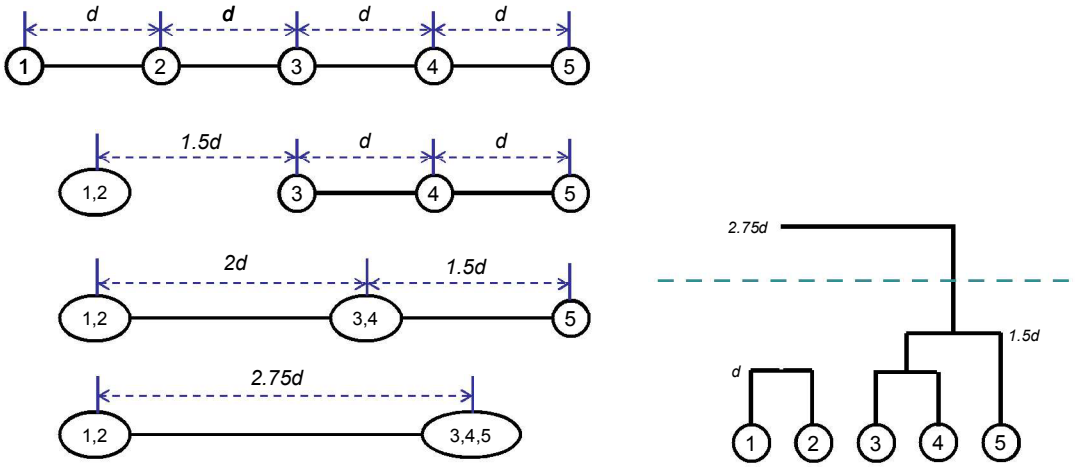


図 5.18　GROUP-SIZE を指定しない階層的クラスタリング法で等間隔図形のクラスタリング過程

Fig5.18　Clustering process of regular interval figures by hierarchical clustering method without GROUP-SIZE.

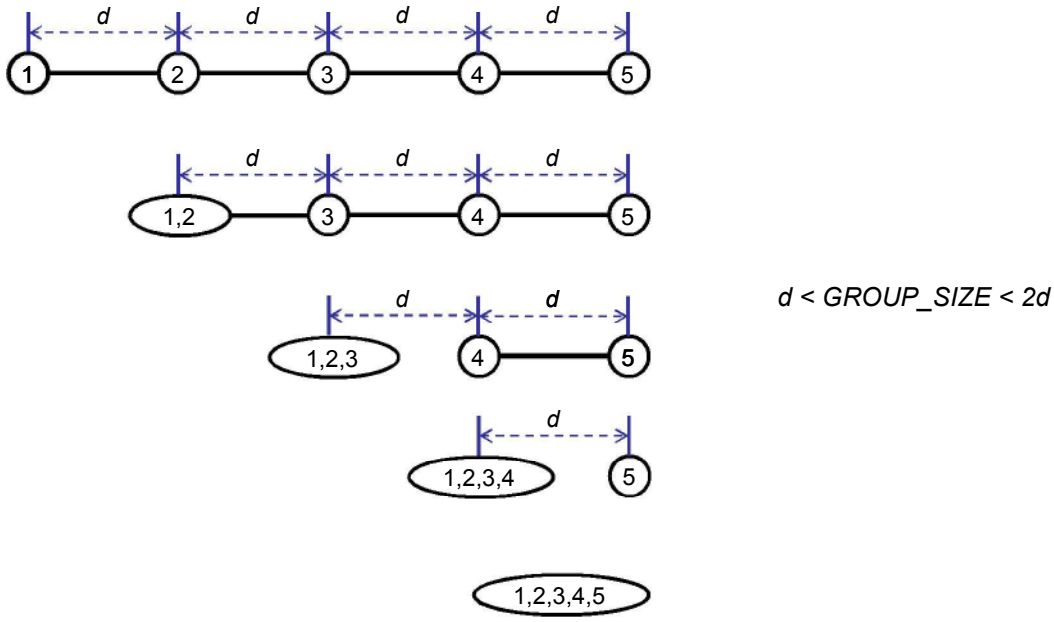


図 5.19　GROUP-SIZE を指定した単リンク拡張法で等間隔図形のクラスタリング処理

Fig5.19　Clustering process of regular interval figures by single link extension method with GROUP-SIZE.

5.5 実験結果

5.4.2 で示したデータにて単リンク拡張法による処理時間の計測を行った. 図 5.20 に示す. クラスタリングの総処理時間を実線で，総処理時間のうちエラー図形の中点算出処理の時間を点線で示している．図 5.15 に示した階層的クラスタリング法での結果に比べて大幅に高速化されかつ，ほぼ線形時間で処理可能になった．例えばエラー図形数 1,200 の場合，従来手法では 30 秒を要していたが（GROUP-SIZE を指定した場合）提案手法では 1.3 秒ほどでクラスタリングが可能になった．さらに，提案手法ではクラスタリング処理の大部分を中点算出処理が占めることがわかった．また出力される画像数（図 5.21 ）および平均の画像サイズ（図 5.22 ）を新旧のアルゴリズムで比較した．5.4.5 で述べたように，単リンク拡張法は，より図形をまとめて大きなクラスタを作成する傾向があるため，同じ GROUP-SIZE の設定であれば階層的クラスタリング法に比べて全体の出力画像数は若干少なくなり，平均の出力画像サイズは大きくなっている.

さらに，5.4.3 で述べた単リンク拡張法における時間計算量の議論を裏付けるため，GROUP-SIZE を変化させて(Step.4) での演算数と要した時間を計測した．図 5.23 に GROUP-SIZE と演算数の関連を示す．演算数とはエラー図形座標と BOX 領域座標の比較回数である．図 5.24 に GROUP-SIZE と処理時間の関連を示す．実験データのチップサイズは 1280um x 1290um なので，GROUP-SIZE を 1300um に設定すると BOX 領域内にすべてのエラー図形が含まれる状態になる．その際は演算数，処理時間ともに線形に推移している．さらに GROUP-SIZE を小さくしていくと線形から乖離していき，1um の設定では，どちらも二乗曲線に近い値になっている.

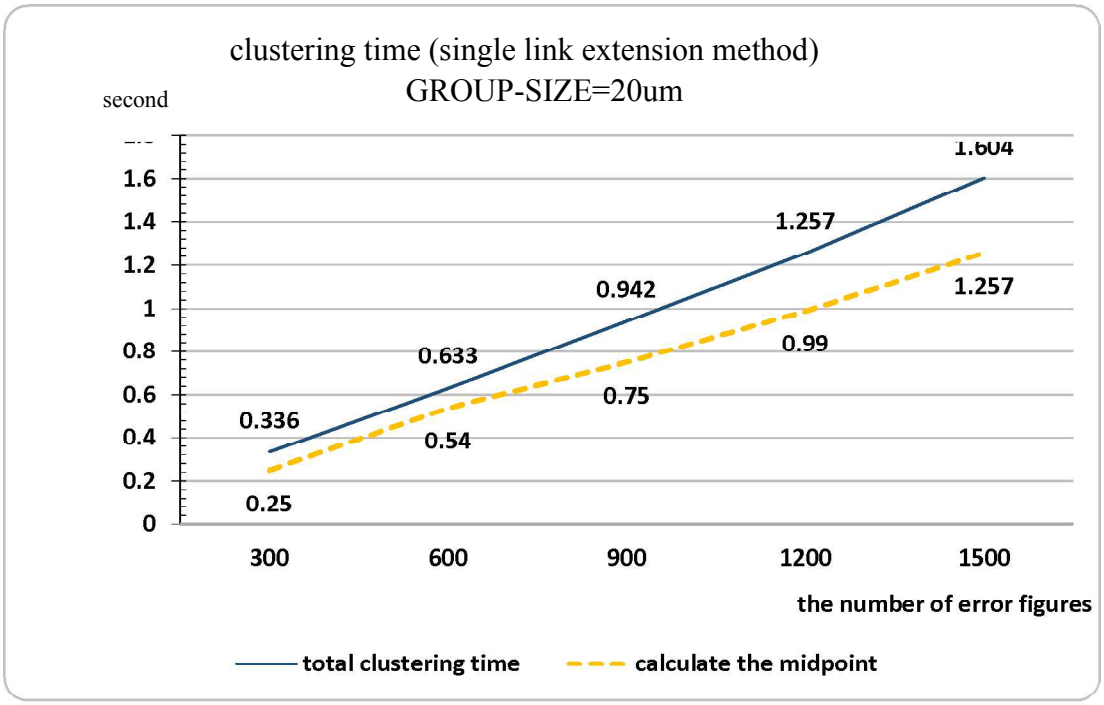
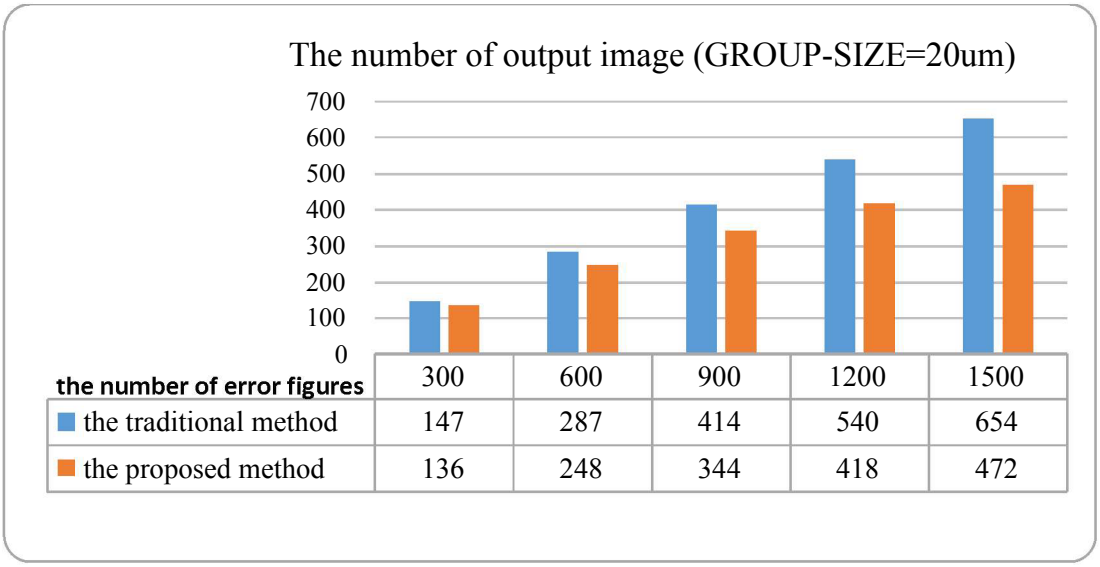


図 5.20　単リンク拡張法での総クラスタリング処理時間とエラー図形の中点を求める時間

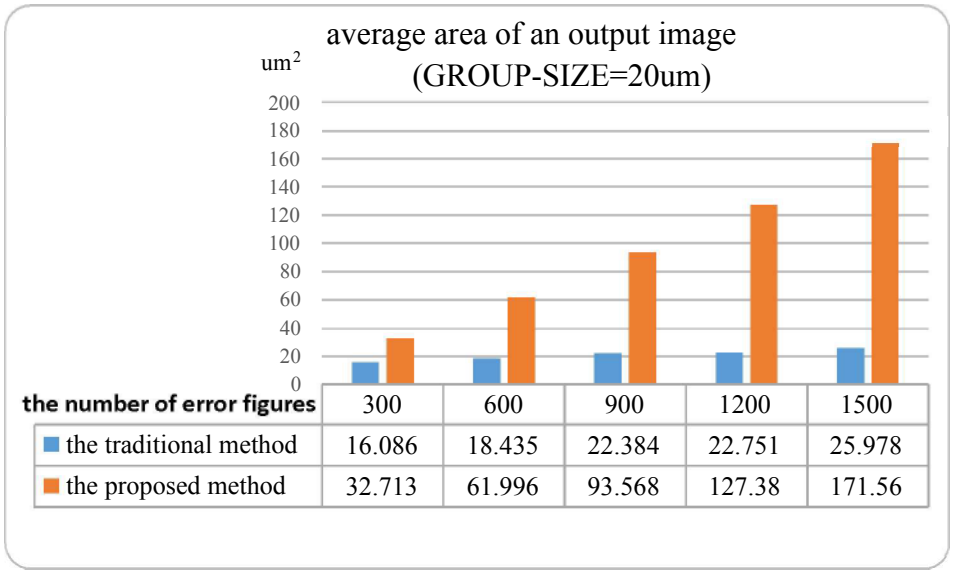
Fig5.20　Processing timeof total clustering and calculating the midpoint of each error figure by single link extensionmethod.



the number of error figures	300	600	900	1200	1500
the traditional method	147	287	414	540	654
the proposed method	136	248	344	418	472

図 5.21　出力画像数（従来手法：階層的クラスタリング法，提案手法：単リンク拡張法）

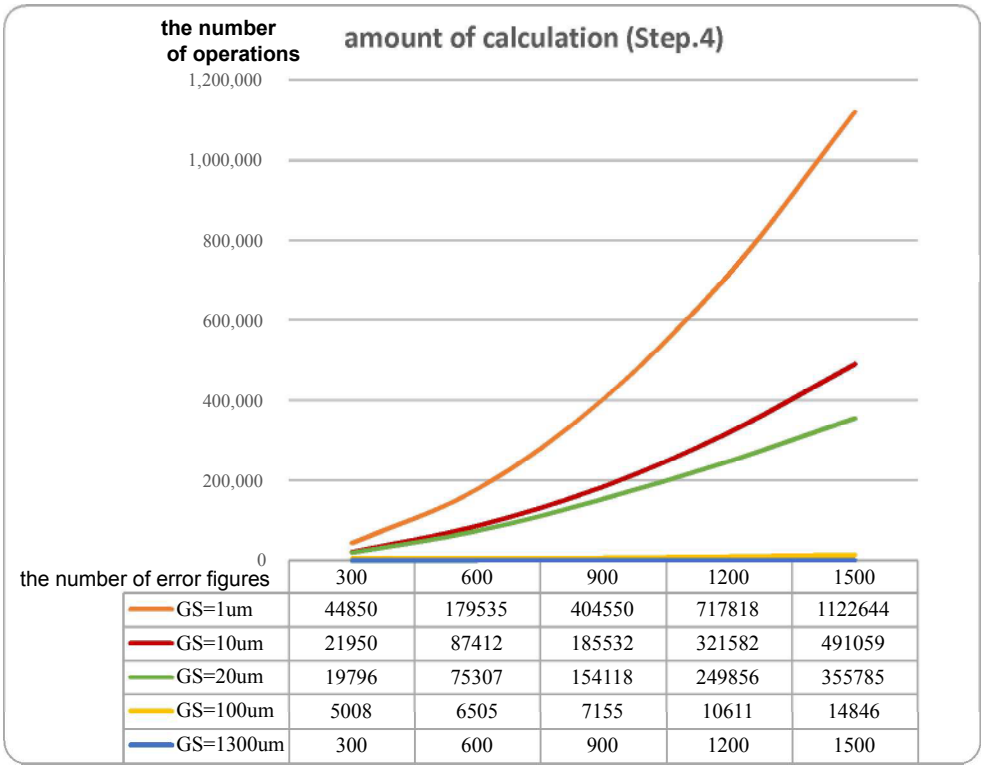
Fig5.21　The number of output image (the traditional method : Hierarchical Clustering ,the proposed method: Single link extension).



the number of error figures	300	600	900	1200	1500
the traditional method	16.086	18.435	22.384	22.751	25.978
the proposed method	32.713	61.996	93.568	127.38	171.56

図 5.22　出力平均画像面積（従来手法：階層的クラスタリング法，提案手法：単リンク拡張法）

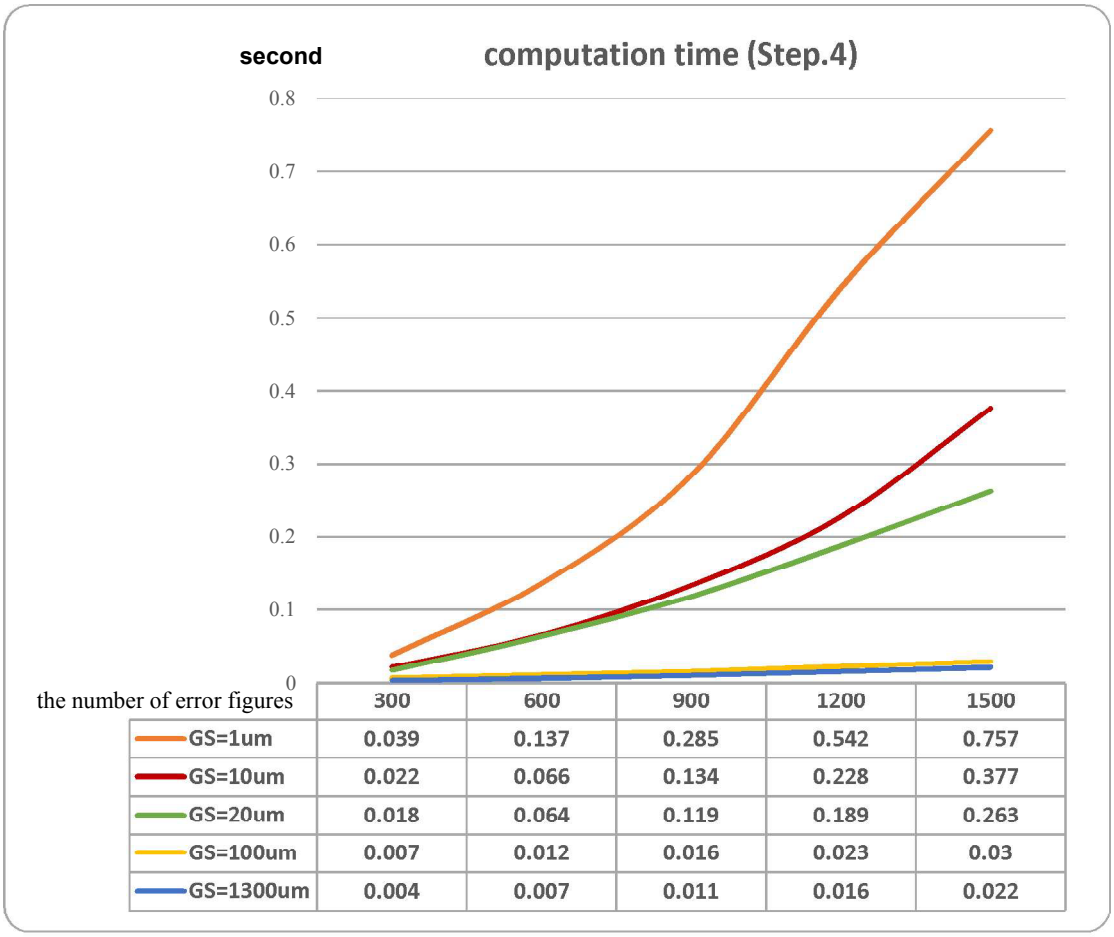
Fig5.22　averagearea of an output image.(the traditional method : Hierarchical Clustering ,the proposed method: Single link extension)



the number of error figures	300	600	900	1200	1500
GS=1um	44850	179535	404550	717818	1122644
GS=10um	21950	87412	185532	321582	491059
GS=20um	19796	75307	154118	249856	355785
GS=100um	5008	6505	7155	10611	14846
GS=1300um	300	600	900	1200	1500

図 5.23　単リンク拡張法 Step.4 での計算量

Fig5.23　The amount of calculation (Step.4 in single link extension method).



the number of error figures	300	600	900	1200	1500
GS=1um	0.039	0.137	0.285	0.542	0.757
GS=10um	0.022	0.066	0.134	0.228	0.377
GS=20um	0.018	0.064	0.119	0.189	0.263
GS=100um	0.007	0.012	0.016	0.023	0.03
GS=1300um	0.004	0.007	0.011	0.016	0.022

図 5.24　単リンク拡張法 Step.4 の処理時間

Fig5.24　The computation time (Step.4 in single link extension method).

5.6 結論

本章では，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．下位セルでのエラーを TOP セルから俯瞰する画像の作成も含めて，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．また，バッチ処理上においてエラー図形を“ある程度のまとまり”ごとに画像保存するために，階層的クラスタリング法と単リンク拡張法を併用したシステムとした．階層的クラスタリング法は GROUP-SIZE を設定することなくクラスタ分割が行えるという利点があるものの，処理時間がおよそ $O(n^3)$かかることが問題であった．一方，単リンク拡張法は概ね線形時間でクラスタリングの処理が可能になり，出力結果も階層的クラスタリング法と比較して，ほとんど問題ないものになった．ただ唯一の懸念点としては，単リンク拡張法では GROUP-SIZE 以内の距離で存在するエラー図形は一つのクラスタになってしまい，比較的大きなサイズのクラスタを作る傾向がある．このことから，現在本システムでは両方のアルゴリズムを選択可能にしている．

今後の検討課題を以下に記す．第一に 5.4.3 の(Step.4) で述べたアルゴリズムにおいて，実装方法を検討し，さらに高速化を図る．第二に GROUP-SIZE 値を作業者の経験に頼ることなく求める機能を実装する．具体的にはいくつかの標本データでチップサイズ，チップのプロセス，DRC エラー図形数を説明変量とし，GROUP-SIZE を目的変量とした重回帰分析を行い，回帰方程式を求めておく．この際の GROUP-SIZE 値は実際に手動で試行した結果から最適と思われる値を与える．そして求められた回帰方程式に実際のデータの情報を代入し，GROUP-SIZE を求める．第三に DRC ツール実行時の条件作成やツール実行そのものの完全バッチ化である．本システムの使用により，DRC ツール実行後の確認作業は自動化された．今後 DRC 検証完全自動化のための課題としては，DRC ツールを実行するまでの作業者の負担減である．デザインルールマニュアル[44] の様式が不定のため条件作成のバッチ化は今のところ難しいが，DRC ツール実行から本システム起動までをバッチ化することは十分可能である．なお本システムは現在，半導体設計事業会社数社にて使用されている．

第 6 章　結論

本研究により得た主たる成果を以下にまとめる．

6.1　研究のまとめ

本研究により得られた主な成果を以下の項目（1）～（4）としてまとめる．

（1）ビアの削減

電源ネットのビアを擬似的な削減方法を検討した．提案の手法により，配線幅チェックに要する時間が大幅に短縮された．得られた結果は次の通りである．

(a) データの大規模化, 特にビア図形の増大により EDA ツールでの解析に時間を要している．ビア図形数の抑制が必要である．

(b) データベース上でビア図形を削減する方法を提案した．

(c) 上記 b により，データベースファイルサイズの削減，データベース読み込み時間の削減，データベース読み込みメモリの削減を確認した．

(d) 一方，データベース作成時間は増加した．これはビア削減処理に要する時間のためである．ただしデータベース作成は 1 データにつき 1 回でよい．

(e) 配線幅チェックの時間は大幅に減少した．数十秒かかっていたものがすべて瞬時に終了するようになった．(d) の増加分を考慮しても十分実用的であるといえる．

（2）電源ネットの枝刈り

電源ネットに Startpoint　と Endpoint　を指定して 2 点間の経路のみを抽出する方法を検討し，有効性を確認した．得られた結果は次の通りである．

(a) グラフ上の任意の 2 点間で経路抽出するアルゴリズムを提案した．処理時間は頂点数＋エッジ数に比例する．

(b) 上記 a を EDA ツール上に実装して配線幅チェックを行った．その結果，電源ネットの幹線部分のみをチェックすることが可能になった．

（3）バッチ処理

（1）および（2）を統合し，Startpoint　と Endpoint　を LVS の結果から自動的に取得するバッチ的な配線幅チェックシステムを作成した．得られた結果は次の通りである．

(a)LVS　のクロスリファレンスファイルを利用して，レイアウトデータ(GDSII) 上で配線幅チェックを行う Startpoint　と Endpoint　の座標を取得できることが示された．

(b)1 および 2 をシステムに実装し，有効性を確認した．

(c)20 か所の配線幅チェックがトータルで 30 分弱で終了した．人手で 2 点間を決定するあるいは擬似エラーを取り除く手間を考えると，大幅な TAT の削減となった．

（4）DRC エラー結果の画像保存システム作成

DRC 実行結果をレイアウトデータに重ねた画像を自動保存するシステムを作成した．得られた結果は次の通りである．

(a)バッチ処理により，DRC ツールの出力を自動的にレイアウトデータ上に重ねあわせて画像保存を行うシステムを作成した．

(b)下位階層のセルで検出されたエラーもトップセル上から俯瞰したイメージで画像保存を行えるようにした．

(c)エラー画像を「適度なまとまり」ごとに保存できるようになった．当初は階層クラスタリング法を採用していたが，エラー図形数が多くなった場合，処理時間に問題が発生することがわかったため単リンク拡張法という高速アルゴリズムを開発した．

（1）~（4）の成果を表 6.1 にまとめる．

表 6.1　本研究の成果まとめ

Table 6.1Theresults summary of this stydy

項目	本研究適用前	本研究適用後
(1)配線幅チェックビア削減	配線幅チェック時間最大約90秒 (表3.3)	配線幅チェック時間1秒未満 (表3.3)
(2)配線幅チェック電源ネット枝刈り	擬似エラー多数出現(図1.5および図3.3左)	擬似エラー無し(図1.5および図3.3右)
(3)LVS結果を利用した配線幅チェックバッチシステム	作業者が検証仕様書を見ながら1か所ずつチェック箇所を指定．擬似エラーは目視で除去 作業時間　1日~2日	自動的にチェック箇所を算出．テーパリングによる擬似エラーは発生せず 作業時間　30分
(4)-(b)DRC結果画像保存・インスタンス配置情報を考慮してTOPセルに重ね合わせ	下位セルでのエラーは当該セルにおいての画像取得になり，当該セルがインスタンス配置されたときに，周りの配線は含まれない (図5.8)	下位セルでのエラー図形をTOPセルから見た画像取得が可能．上位セルに存在する配置図形も合わせて確認できる (図5.9)
(4)-(c)DRC結果画像保存・クラスタリング処理	エラー図形ごとに画像取得し，大量の画像が保存される(図5.1右上)	エラー図形のまとまりごとに画像取得し，画像の数は限定される(図5.1右下)

結論として，提案した手法およびそれに基づいて作成したシステムは LSI の大規模化と複雑化に伴って顕著になってきた課題を解決し，高いパフォーマンスをあげることができた．上述の（1）～（2）は LAVIS に実装された後，LAIVS-plus に機能が移行され，現在もアナログ IC の設計部門を中心に利用されている．（4）も複数の半導体設計事業会社で利用され，設計・検証の効率向上に効果があることが実証されている．

6.2　今後の課題

今後の課題として，以下の点が挙げられる．

(1) 本研究ではレイアウトデータの削減のためにビアを削減することが効果的であった．しかし，レイアウトデータから電気的特性を抽出するような処理の場合，たとえば，ビアを含む配線ネットの抵抗値を算出したい場合，ビアの削減を行うと配線の抵抗値が変わってしまう．DRC の配線幅チェックにおいてはビア削減は問題ないが，配線上の 2 点間の抵抗値を求めたいときにはビアの削減は行えない．抵抗値計算の処理時間のボトルネックは多数のビアが置かれている場合になっており，ビアを考慮しながら高速に計算処理する手法が必要である．

(2)現在，2 点間を指定してネットの枝刈りをする場合にビアの配置と配線幅違反の位置関係によっては，違反図形として出力できないｹｰｽがある（図 6.1）．これは配線 AND 領域を結果図形に OR して出力することで解決できる．ただし，処理時間は増加することになる．

(3)DRC エラー図形の画像保存において，新たに開発した単リンク拡張法の実装上の技術として，現在採用しているデータ構造よりも高速に行える方法（Range Tree)[52][53] が存在することがわかっている．高速化については，常に改善が要求されているため今後も最適な手法を継続的に検討し，導入していくことが必要である．

(4)DRC エラー図形の画像保存において，GROUP-SIZE 値を作業者の経験に頼ることなく求める機能を検討する．具体的にはいくつかの標本データでチップサイズ，チップのプロセス，DRC エラー図形数を説明変量とし，GROUP-SIZE を目的変量とした重回帰分析を行い，回帰方程式を求めておく．この際の GROUP-SIZE 値は実際に手動で試行した結果から最適と思われる値を与える．そして求められた回帰方程式に実際のデータの情報を代入し，GROUP-SIZE を求める手法を考えている[55][56][57] ．

(5)DRC ﾂｰﾙ実行時の条件（ﾙｰﾙ）作成やﾂｰﾙ実行そのものの完全バッチ化である．5 章で述べたシステムの使用により，DRC ツール実行後の確認作業は自動化された．今後 DRC 検証完全自動化のための課題としては，DRC ツールを実行するまでの作業者の負担減があげられる．デザインルールマニュアル[44]の様式は製造プロセスが異なれば様式も異なるし，同じプロセスであっても設計事業会社や社内の部門によって異なる場合がある．このため，条件（ルール）作成のバッチ化は今のところ難しいが，LAVIS-plus から DRC ツールを起動してその結果を自動的に取得することにより，DRC ツール実行から本システム起動までをバッチ化することは十分可能である．

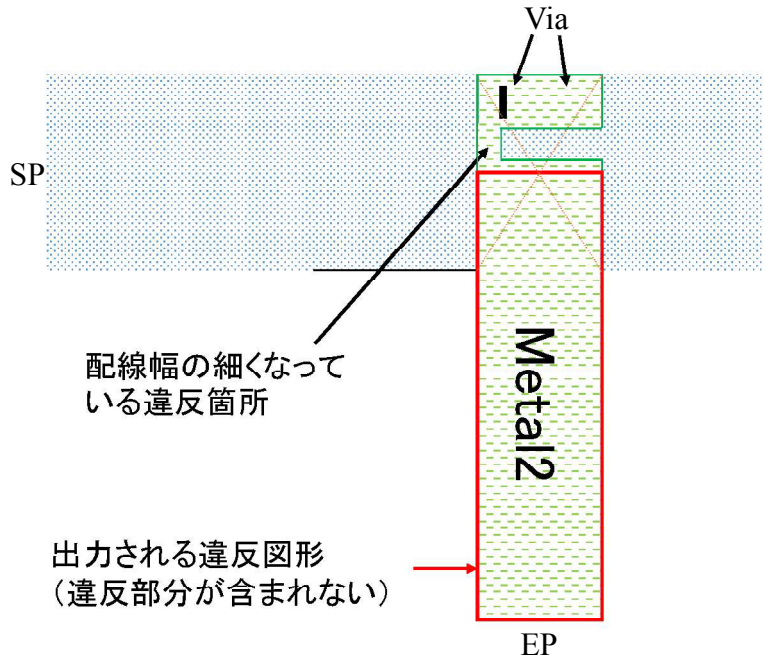


図 6.1　エラー図形が正しく出力されないケース

Fig.6.1　The case by which an error figure isn't output right

最後に，本論文の内容は，筆者が TOOL 株式会社 EDA 事業部に在職中（2007 年 4 月から現在まで）ならびに早稲田大学大学院情報生産システム研究科情報生産システム工学専攻博士後期課程に在学中（2011 年 4 月から現在まで）に行ってきた「大規模 LSI における EDA ツールの性能向上と改良に関する研究・開発」をまとめたものであることを付記する．

参考文献

[1]International Technology Roadmap for Semiconductors 2011 Edition, Semiconductor Industry Association, Executive summary.

[2] 西久保靖彦，基本システム LSI 用語辞典，CQ 出版（株），東京，2000.

[3] S. Leitner and H. Wang, “Current compensation techniques for low-voltage high-performance current mirror circuits,” Analog integrated circuits and signal processing, vol.88, pp.79-88, 2016.

[4] S. Shim, W. Chung and Y. Shin, “Lithography Defect Probability and its Application to Physical Design Optimization,” IEEE transaction on very large scale integration (VLSI) systems, 99, pp.1-15, 2016.

[5] 児玉親亮，中山幸一，小谷敏也，野嶋茂樹，三本木省次，宮本晋示，“超微細化を実現する側壁ダブルパターニングのためのレイアウト設計手法，” 情処学研報，2011-SLDM-153(25)，pp.1-6，2011.

[6] 伊藤俊明，鹿本俊幾，小谷憲，清水良浩，寺井正幸，“ブリッジ故障対応 IDDQ テストで用いる近接配線ペアリストの効率的生成手法，”情報処理学会論文誌 48(5)，pp.1898-1905，2007.

[7] 吉岡信行, DFM 技術,”LSI テスティングハンドブック, LSI テスティング学会(編), pp134-139, オーム社, 東京, 2008.

[8] 今井正治（監修），改訂版 EDA 用語辞典，（社）電子情報技術産業協会 EDA 技術専門委員会，日経 BP 社，2008

http://techon.nikkeibp.co.jp/article/WORD/20081219/163104/

[9] D.Y. Hu and C.Z. Chen, “Reliability Aspects of Advanced IC Technology with ESD and

Anti-Radiation Capabilities," ECS transactions, vol.60, pp.1185-1190, 2014.

[10]Yuji.Takashima, Kazuyuki Oo ya, and Atsushi Kurokawa, "Practical Redundant-Via Insertion Method Considering Manufacturing Variability and Reliability," IEICE Trans. Fundamen-tals, vol.E92-A, no.12, pp.2962-2970, Dec.2009 .

[11]Cheok-Kei Lei, Po-Yi Chiang, Yu-Min Lee, "Post-Routing Redundant Via Insertion with Wire Spreading Capability," Asia South Pacific Design Automation Conference (ASPDAC), pp. 468-473, Jan.2009.

[12] S.K. Nandy, "Geometric design rule check of VLSI layouts in distributed computing environment," VLSI design, vol.1(2), pp.155-167, 1994.

[13] 伊藤則之，安永守利，"プロセッサ設計手法の現状と今後－高性能化を実現する設計フローと CAD システムー，" 信学論(D), vol.J94-D, no.12, pp.2004-2030, Dec. 2011.

[14] 川上善之，草野健次，寺尾誠，石嶋宏亘，福井正博，"製造不確実時代における回路のディペンダブル動作を保障する電源配線最適化手法，"情報処理学会論文誌 49(6), pp.2144-2154，2008.

[15] R. Goering, "Design rule check, layout-vs.-schematic tools roll," Electronic Engineering Times, 1388, pp.31-47, 2005.

[16] P. Shreepad, K. Samadi, Y. Du and S.K. Lim, "Design and CAD methodologies for low power gate-level monolithic 3D ICs," ISLPED'14, pp.171-176, 2014.

[17]S.M.Rubin, Computer Aids for VLSI Design, 2nd ed., Addison-Wesley Publishing Company, California,1994.

[18] LAVIS.Users-Guide,TOOL Corp., 2011

[19] LAVIS-plus.Users-Guide,TOOL Corp., 2015

[20] Open Artwork System Interchange Standard (OASIS), SEMI P39-0304E San Jose: SEMI 2004, 2008, http://www.semi.org/en/

[21] R.R. Pai, "Oasis comes up short as GDSII replacement," Electronic Engineering Times, no.1463, pp.52-54, Feb. 2007

[22] M. Lutz, Programming Python Fourth Edition, O'Reilly Media, Inc., 1005 Gravenstein

Highway North, Sebastopol, CA95472, 2011.

[23] J.K. Ousterhout, and K. Jones, Tcl and the Tk Toolkit second Edition, Pearson Education, Inc., 501 Boylston Street,Suite900 Boston, 2010.

[24] D. Flanagan, and Y. Matsumoto, The Ruby Programming Language, O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA95472, 2008.

[25]LAVIS.CommandReference.pdf, TOOL Corp., 2012.

[26] P. Chen, D.A. Kirkpatrick, and K. Keutzer, "Scripting for EDA tools: a case study," International Symposium on Quality of Electronic Design, March 2001.

[27] Dan Clein, "CMOS IC Layout Concepts, Methodologies and Tools", Newnes, Elsevier,2000.

[28] S. Anandakumar, "Computationally Efficient Simulation of High-Frequency Transients in Power Electronic Circuits," IEEE transactions on power electronics, vol.31, no.9, pp.6351-6361, 2016.

[29] 平井一寛，亀井智紀，"LSI 設計支援ソフトウェア「LAVIS」による設計エラー解析，"第 29 回 LSI テスティングシンポジウム，no.C1, pp.129-133，大阪，Nov. 2009.

[30] S.E.Shulze, and G.E.Bailey, "Distributed processing in integrated data preparation flow," Proc. SPIE 5567, 24th Annual BACUS Symposium on Photomask Technology, no.43, pp.394-405, Monterey, CA, Dec. 2004.

[31] 福本尚人，中島耕太，成瀬彰，"メニーコア・プロセッサにおける動的スレッド切替え手法，" 情処学研報（ハイパフォーマンスコンピューティング）,2013-HPC-140(34), pp.1-6, Jul 2013.

[32]Calibre Standard Verification Rule Format (SVRF) Manual, Mentor Graphics.,2013.

[33] 亀井亮児，"CentOS 5.6 Red Hat 互換の安定した OS，"日経 Linux13(8)，日経 BP 社，pp.71-73，2011.

[34] G. Radhika, "Block-Level Electro- Migration Analysis (BEMA) for Safer Product Life," VLSI design, 2015-Feburary, pp.276-281, 2015.

[35] J. Palkesh, "Design-in-reliability:From library modeling and optimization to gate-level

verification," Microelectronics and reliability, vol.54, no.6-7, pp.1421-1432, 2014.

[36] R. セジウィック，アルゴリズム　第 3 巻＝グラフ・数理・トピックス，近代科学社，東京，1993.

[37] 安藤清，土屋守正，松井泰子，例題で学ぶグラフ理論，森北出版，東京，2013.

[38] 築山修治，“アルゴリズムとデータ構造の設計法”，pp.133-136，コロナ社，東京，2003

[39]Calibre nmDRC/nmLVS　　簡易リファレンス　Software Version 2008.4, Mentor Graphics.,2009.

[40]Virtuoso.Users-Guide, Cadence Design Systems,Inc., 2011.

[41]lvgds2pilot.txt, TOOL Corp., 2011.

[42] 川中子敬至，“多重な内包のある多角形群に与えられた点の内外判定とその応用，”日本経営工学会論文誌 57(3)，pp.253-259，Aug 2006.

[43]F.P. Preparata, and M.I. Shamos, Computational Geometry: An Introduction, Springer, 1985.

[44] 名倉徹，LSI 設計常識講座，東京大学出版会，東京，2011

[45]lverrdb2gds.txt, TOOL Corp., 2014.

[46] 平井一寛，高橋利和，“ソフトウェア「OASIS-Utility 」，「LAVIS」を用いた故障解析支援，”　第 30 回 LSI テスティングシンポジウム，no.C6, pp.131-135 ，大阪，Nov. 2010.

[47] 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

[48] 宮本貞明，クラスター分析入門，森北肇，森北出版（株），東京，1999.

[49] 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”，情処学研報，2003-CVIM-141，pp.57-62 ，2003.

[50]A. Bouguettaya, Q. Yu, X. Liu, X. Zho u and A. Song, “Efficient agglomerative hierarchical clustering,” Expert Systems with Applications, 42, pp.2785-2797, 2015.

[51] B. Chandramouli, and J. Goldstein, “Patience is a Virtue: Revisiting Merge and Sort on

Modern Processors," "SIGMOD/PODS'14", pp.731-742, Salt Lake City, UT, USA, 2014.

[52] M. ドバーグ，M. ファン・クリベルド，M. オーバマーズ，O. シュワルツコップ，コンピュータ・ジオメトリー計算幾何学：アルゴリズムと応用－，浅野哲夫，（株）近代科学社，東京，2000.

[53] R. セジウィック，アルゴリズム　第 2 巻＝探索・文字列・計算幾何，近代科学社，東京，1993.

[54] 長畑秀和，多変量解析へのステップ，共立出版（株），東京，2001.

[55] 岩崎学，統計的データ解析入門 単回帰分析，東京図書（株），東京，2006.

[56] 吉岡信行，“DFM 技術,” LSI テスティングハンドブック，LSI テスティング学会（編），pp134-139，オーム社，東京，2008

[57] 涌井良幸，涌井貞美，図解でわかる回帰分析，中村洋一郎（編），（株）日本実業出版社，東京，2002.

発表関連論文及び資料

学術誌原著論文

1. 亀井智紀，渡邊孝博，“LSI の効率的な DRC検証作業の支援システム”，電子情報通信学会論文誌，Vol.J99-D ， No.6，pp.607-616 ， 2016 年 6 月.

2. 亀井智紀，川北真裕，渡邊孝博，“LVSの出力情報を活用した VLSI電源配線幅の高速検証システム，”電子情報通信学会論文誌，vol.J96-D ，no.5，pp.1330-1337 ，2013 年 5 月.

国際会議（査読付き）

3. Tomoki Kamei and Takahiro Watanabe, “Automatic Viewing System for a Vast Number of DRC Errors using Agglomerative Hierarchical Clustering, ” ITC-CSCC2015, Seoul, Korea, no.OS24-3, pp.699-702, July 2015.

4. Tomoki Kamei and Takahiro Watanabe, “A Generation of Cross-Section Views based on LSI Layout Data for Efficient Failure Analysis, ” ITC-CSCC2013, Yeosu, Korea, MB1, pp.305-308, July 2013.

5. Tomoki Kamei and Takahiro Watanabe, “Rotational Display Problem for Array Reference LSI Layout Data,” ITC-CSCC2012, Sapporo, Japan, C-T3-06, pp.-. July 2012.

6. Tomoki Kamei, Masahiro Kawakita and Takahiro Watanabe, “The High-speed power Line Topology Check by Reducing Vias, ” 48th Design Automation Conference, San Diego, USA, 10U, Poster, June 2011.

国内会議（査読付き）

7. 亀井智紀，渡邊孝博，“DRC検証作業の負荷を軽減するシステムの開発，”FIT2015

第 14 回情報科学技術フォーラム，松山，第 1 分冊, no.RC-009, pp.69-74, 2015 年 9 月.

研究会

8. 亀井智紀，安部拓哉，本垰秀明，渡邊孝博，“Via 数削減による大規模レイアウトの高速 DRC手法，”情報処理学会研究報告. SLDM，システム LSI 設計技術 2011-SLDM-148(17), pp.1-6, 2011 年 1 月.

電子媒体記事

「【DAC 201】TOOLと東芝，早大が共同で，マスク・レイアウト検証の電源配線チェックを効率化」，日経テクノロジーonline ，小島郁太郎，2011 年 6 月

http://techon.nikkeibp.co.jp/article/NEWS/20110615/192606/