ノイズ耐性を持つ電源/GND ネットワークの最適化設計手法

Power/Ground Networks Optimization Design Methods with Noise Immunity

2012 年 2 月

田中　美樹子

ノイズ耐性を持つ電源/GND ネットワークの最適化設計手法

Power/Ground Networks Optimization Design Methods with Noise Immunity

2012 年 2 月

早稲田大学大学院　基幹理工学研究科

情報理工学専攻　マルチメディアシステム研究

田中　美樹子

Table of Contents

Chapter 1

序論

1.1 本論文の背景と目的

近年、プロセス技術の進歩に伴い、回路の動作速度は向上した。また、搭載されるデジタル回路規模が増加し、これに伴い大電流化が進み、デジタル回路におけるノイズ、即ち電圧降下（電圧ドロップ）は増大の傾向にある。一方、低消費電力化の要求によって、低電圧化が進んでいる。このため、相対的にノイズが大きくなり、デジタル回路のノイズ耐性は弱くなる傾向にある。デジタル回路に於いては、電圧ドロップによる誤動作や遅延劣化が原因の動作速度低下が問題となっている。これは、高速化要求への障害となっている。

一方、ノイズに敏感なアナログマクロ（HDMI、 DDR、 USB 等高速 IO マクロ、 ADC、DAC マクロ、アンテナマクロ）とデジタル回路が 1 チップに搭載されるようになってきた。これによって、アナログマクロの誤動作や特性劣化が顕著化してきている。ノイズ問題は、一旦発生すると解決に多くの時間を要することが多く、大きな問題となっている。回路が放出したノイズをその回路自身が受け誤動作を引き起こす自家中毒は、 LSI を製造するメーカにとっては、最も考慮すべきノイズである。また、近年、 EMC では、 VCCI 等の規格が制定され、規格を満足しないシステムの販売が禁止されている。さらに、ノイズイミュニティ (ノイズ耐性) に対する要求も増大している。加えて、近年の価格競争の激化によって、低コスト化の要求が増大している。 LSI のサイズは、コストに直結するため、コスト削減に於いて、最も重要な項目である。このため、チップ内で大きな面積を占めている、電源 /GND ネットワークやオンチップデカップリング容量削減は重要な項目である。また、設計期間の短縮要求によって、高速に、精度良く解析、設計を行える設計手法が重要となってきている。これらの要因から、正確な電圧ドロップ、基板ノイズの解析モデルの構築が重要となってきている。モデル化に於いて、以下の 4 つが重要な項目である。

1） ノイズ（電圧ドロップ）の発生を正しくモデル化

２） 電圧ドロップによる遅延変動を正しくモデル化

３） 伝播経路を正しくモデル化

４） ノイズを受けた場合のアナログマクロの動作を正しくモデル化

そこで、本論文では、これらの 4 つを正確にモデル化し解析を行う方法、即ち、マクロモデリングによるノイズ解析アルゴリズムを提案する。本アルゴリズムは、非常に高速であるため、最適化アルゴリズムのノイズ耐性チェックとして有効である。また、コスト削減のために、ノイズ耐性を満足し、かつ電源 /GND ネットワークの総配線面積、オンチップデカップリング容量の総面積を最小化することが重要である。加えて、アナログマクロに伝播する基板ノイズをチップサイズ、配線層数の増加なしに、アナログマクロのノイズ耐性以内となる様に減少させることも重要である。これらを実現するために、本論文では、上で述べたマクロモデリングによるノイズ解析アルゴリズムを適用し、高速に高精度に電源 /GND ネットワークの最適化設計を行う以下の 3 つのアルゴリズム：

１） 電源 /GND ネットワーク最適化アルゴリズム

２） オンチップデカップリング容量最適化アルゴリズム

３） 基板ノイズ最適化アルゴリズム

の提案を行う。

1.2 本論文の概要

本論文は以下の各章から構成される。

第 2 章では，大規模回路に対して、電圧ドロップ、基板ノイズ解析を高速に、高精度に行う方法について議論する。デジタル回路のタイミングマージンを悪化させ、アナログマクロの特性を劣化させる電圧ドロップ、基板ノイズを高速に、高精度に解析することは、回路が正しく動作するかどうかを判断するために重要な課題である。 Badaroglu は、プリミティブブロックレベルでマクロモデリングを行い、電圧ドロップ、基板ノイズ解析を行う方法を提案した (IEEE trans. 2006)。この方法は、トランジスタレベルで解析を行うのではなく、プリミティブブロックレベルでモデル化を行い、解析を行うため、トランジスタレベルの解析アルゴリズムと比べ高速化が期待できる。しかし、 SoC の様な大規模回路に於いては、処理時間、

メモリサイズの面で適用することは困難である。そこで、提案アルゴリズムは、大規模回路を高速に、精度良く解析する為に、 LSI を複数のドメインに分割し、そのドメイン毎にマクロモデルを作成し、電圧ドロップ解析、基板ノイズ解析を行う。ドメイン数を削減することによってノード数が減少するので、計算複雑度は大きく減少する。しかし、解析精度は悪化する。そこで、ドメインサイズ、解析時間、解析精度のトレードオフ関係を考察した。考察結果から、ドメインの電流密度変動係数と見積もり基板ノイズ誤差の間に線形の関係があることが判明した。この結果から、ドメインサイズをドメインの電流密度変動係数を元に決定することによって、精度と処理時間の両方を満足できることが判明した。提案アルゴリズムでは、この線形関係を利用し、まず目標誤差を設定し、その誤差に対応する電流密度変動係数を求め、各ドメインを電流密度変数係数以内となるようにドメイン分割を行う。これによって、精度と解析時間の両立を実現する。実験結果から、ノイズ誤差を 15 % に設定し、ドメイン分割を行った場合、見積もりジッタ（クロック周期のゆらぎ）の精度は 85% 以上であった。これは、電流密度変動係数による、ドメイン分割が正しく動作していることを示す。また、提案アルゴリズムは Badaroglu のアルゴリズムと比べ、 10000 倍以上高速であった。提案アルゴリズムを適用すると解析速度と精度の両方を満足できるので、数千億のトランジスタを持つマイクロプロセッサの様なシステム LSI に対しても適用可能な方法である。また、ノイズ耐性のチェックアルゴリズムとして最適化アルゴリズムで適用することが可能な有効な方法である。

第 3 章では、電圧ドロップ制約を満足し、同時に電源 /GND ネットワーク配線の総面積を最小化するアルゴリズムについて議論する。電源 /GND ネットワーク配線の総面積の増加は、配線性、タイミングクロージャーに大きな影響を与える。配線層数を最小化し、製造コスト、製造 TAT を削減することが、コスト削減の目的から求められている。 Zhao は、この問題を数学的に定式化し、非線形計画問題として解く方法を提案している (DAC 2007)。しかしこの方法では、 SoC の様な大規模回路に適用することは難しい。また、緩和のため、解が悪化する。一方、 Singh は、ノイズ制約を満足しない場合、制約を満足しないドメイン内の配線幅を均一に増加させる欲張り最適化アルゴリズム（greedy optimization algorithms）を提案した (IEEE trans. 2006)。このアルゴリズムの 1 つの欠点は、均一に電源 /GND ネットワークの配線の抵抗を変更するので、電源 /GND ネットワークの配線面積が大きくなることである。提案アルゴリズムは、非線形計画問題を逐次線形計画法で解くのではなく、電流をフロー、許容電圧ドロップ値を容量として捕らえ、ネットワークフロー問題として解くことによって解を求める。即ち、ネットワークの枝の容量を増やす既存ネットワークの最小費用改良問題

の拡張問題として捉え解を求める。これによって、非線形計画問題を解く時に用いる緩和を行う必要がなく、良い解を得ることができる。提案アルゴリズムは、まず、電圧ドロップに影響のあるパスを求める。これは、解析空間を絞り、解析時間の向上に有効である。次に、そのパスの中でフローが容量を最も超えて流れているパスを 1 つ求める。その後、パス上の枝で電圧ドロップに最も影響のある枝を見つけ、その枝の抵抗を変更、即ち容量を増加させることによって電圧を改善する。提案アルゴリズムは電圧ドロップに最も影響のある枝を更新するため効率的な方法である。また、提案アルゴリズムは電流モードではなく電圧モードで制約を設定する方式を採用した。これによって、電圧ドロップ、即ち電圧を直接制約として扱うことが出来るようになり、 Zhao のアルゴリズムと比べ抵抗による電圧誤差を解消でき、良い解を得ることが出来るようになった。計算機実験結果から、 Singh のアルゴリズムと比べ電源 /GND ネットワークの配線面積を 1/3 に縮小することを示した。また、提案アルゴリズムは Singh、 Zhao のアルゴリズムでは電圧ドロップ制約を満足することができなかったデータに対しても電圧ドロップ制約を満足する解を得ることができた。

第 4 章では、電圧ドロップ制約を満足すると同時に、オンチップデカップリング容量の総面積を最小化するアルゴリズムについて議論する。オンチップデカップリング容量の追加は、電圧ドロップを解決する重要な手法である。オンチップデカップリング容量の追加は、チップ面積の増加、リーク電流の増加を招くため、追加する量を如何に減少させるかが重要な課題である。 Zhao は、この問題を数学的に定式化し、非線形計画問題として解く方法を提案している (DAC 2006)。しかしこの方法では、 SoC の様な大規模回路に適用することは難しい。また、緩和のため、解が悪化する。提案アルゴリズムは、非線形計画問題を逐次線形計画法で解くのではなく、電流をフロー、許容電圧ドロップ値を容量として捕らえ、ネットワークフロー問題として解くことによって解を求める。即ち、ネットワークの枝を追加する既存ネットワークの最小費用改良問題の拡張問題として捉え解を求める。非線形計画問題を解く時に用いられる緩和を行う必要がなく、良い解を得ることができる。提案アルゴリズムは、電圧ドロップに影響のあるパスを求め、そのパスの中でフローが容量を最も超えて流れているパスを 1 つ求める。次に、パス上のノードで、電圧ドロップに最も影響のあるノードを見つけ、そのノードにオンチップデカップリング容量を追加、即ちそのノードにソースノードからの枝を追加し、電圧改善を行う。電圧ドロップに最も影響のあるノードにオンチップデカップリング容量を追加するため、効率的なアルゴリズムである。また、オンチップデカップリング容量から電荷が供給される時間を考慮し、電流が必要なノードと電流を供給するオンチップデカップリング容量

の距離を、一定距離以内にする制約を設定し、実効的なオンチップデカップリング容量追加を可能にした。これによって効果のない箇所へのオンチップデカップリング容量の追加を避けることができ、オンチップデカップリング容量の総面積の増加を抑えることができた。計算機実験結果から、提案アルゴリズムは実サイズデータに対し、処理時間が数分と実用的な方法である。また、 Zhao のアルゴリズムと比べ、オンチップデカップリング容量の総面積を 40 〜 50% に減らせることを示した。

第 5 章では、基板ノイズ削減を、デジタル電源 /GND ネットワークを最適化することによって実現するアルゴリズムについて議論する。デジタル、アナログ混載 LSI に於いて、デジタル電源ノイズは、 Si 基板を介してアナログマクロに伝播しアナログマクロの特性を劣化させる。この電源ノイズを、基板ノイズと呼ぶ。アナログマクロの特性劣化を防止するためには、基板ノイズの伝搬量をアナログマクロのノイズ耐性以内にすることが重要である。この問題を解決するために、ガードリング手法（Bronckers, IEEE trans. 2010）が提案されている。ガードリングを用いた従来手法では、ガードリング追加によって、 LSI の面積、製造マスク数が増加し製造コストが増加する。そこで、基板ノイズの原因がデジタル電源ノイズであることに着目し、デジタル電源ノイズ (ノイズ源) を直接削減し、基板ノイズの低減を図るアルゴリズムを提案する。直接デジタル電源ノイズを削減するため、 LSI の面積増加、製造マスク数増加無しに、基板ノイズを削減することができ、有効なアルゴリズムである。提案アルゴリズムは、アナログマクロに最も電流を伝播しているデジタル回路を電流経路探索によって特定する。次に、この最も電流をアナログマクロに供給しているデジタル回路の電圧ドロップを、 3 章で述べた電圧ドロップ改善アルゴリズムを適用し改善を行い、アナログマクロに伝播する電流の総量を削減する。提案アルゴリズムは、ガードリングを用いる従来手法と比べ、 LSI 面積の増加、製造マスク数の増加無しに、基板ノイズを低減できる点で優れている。計算機実験結果から、提案アルゴリズムは、実データサイズデータに対し、処理時間 4 分 6 秒で LSI 面積の増加無しに、 ADC のノイズ耐性以内に基板ノイズを削減する結果を得た。

第 6 章では，本論文全体を総括する。

Chapter 2

マクロモデリングによるノイズ解析アルゴリズム

2.1 概要

スーパーコンピュータで用いるマイクロプロセッサのデザインは、非常に大きな回路規模と高速動作のため、とても挑戦的な設計である。タイミング制約は、このような高速に動作するプロセッサでは、非常に厳しい。そのため、正確なクロックジッタの見積もりが重要である。クロックジッタは、主に PLL の電源ノイズとクロック分配ネットワークが原因である。PLL の電源ノイズはノイズの伝播を防止する目的でデジタルコアの電源と分離されるが、ノイズは、 Si 基板を介して伝播し、 PLL に影響を与える。このノイズを、基板ノイズと呼ぶ。基板ノイズは、論理回路のタイミングマージンを悪化させるだけでなく、 PLL のジッタも悪化させる。タイミングを基準にして設計品質を確保するために、電源ノイズだけでなく、基板のノイズを正確に解析することが不可欠である。

この要求を満足するために、基板ノイズ解析のための方法 [1]- [9]が提案されている。SWAN [2] では、プリミティブブロックレベルマクロモデルを用い、スタンダードセルベースのデザインを回路シミュレーションする方法が提案されている。このアルゴリズムを以下に示す。

1) プリミティブブロックレベルのマクロモデルが作成される。

2) 電源 /GND ネットワークと基板ネットワークの RC モデルが抽出される。

3) マクロモデル、電源電圧源は、電源 /GND/ 基板ネットワークの RC モデルに接続され、回路シミュレーションされる。

これら方法では、扱えるデザインのサイズは RF 回路のように小さなデザインに制限されており、プロセッサーのようなサイズには、適用が難しい。この制限は、主に計算機の処理時間、メモリサイズからきている。即ち、基板ノイズ解析に十分な精度の基板のネットリスト

（RC モデル）を抽出するには、 基板のネットリストのサイズが大きくなりすぎ、実用的な処理時間で基板ノイズを解析することは、難しい。このため、われわれは、解析複雑度を大きく削減し、高速に基板ノイズを解析する方法を開発し、 350M トランジスタのプロセッサデザインに適用をした。提案アルゴリズムは、大規模サイズのチップを複数のドメインに分割する。それぞれのドメインでは、トランジスタ、電源 /GND ネットワーク、基板ネットワークの等価回路が作成され、マクロレベルの回路モデルに縮退される。ゆえに計算複雑度は大きく減少する。ドメインサイズと解析時間のトレードオフ関係を調べた我々の実験結果から、ドメインサイズを大きくすると解析精度が悪化することがわかった。また、ドメインの電流密度変動係数と解析誤差の関係を調査した。調査の結果、ドメインの電流密度変動係数と解析誤差の間には、正比例の関係があることが分かった。この議論から、ドメインの電流密度変動係数を用いてドメインサイズを決定することによって、精度と計算複雑度の両方を満足できることがわかった。

提案の基板ノイズ解析アルゴリズムを、スパーコンピュータで用いるプロセッサ設計に適用した。この設計では、タイミング制約の厳しさから、基板ノイズだけでなく、基板ノイズが原因で発生するクロックジッタも見積もった。そして、実測結果と比較を行い見積もりクロックジッタの精度を検証した。実験結果から、見積もりジッタの精度は、解析精度を誤差 15% に設定した場合、 85% 以上であることがわかった。そして、解析時間は、デザインサイズが 1/3000 であるデザインを従来のプリミティブブロックレベルのアルゴリズム [2]を用いて解析した時間と比べ 5 倍高速であることがわかった。即ち、従来アルゴリズムと比べ、我々のアルゴリズムを用いると、約 10000 倍高速である。我々が提案するアルゴリズムを用いると、解析速度と精度の両方を満足できるので、数百億のトランジスタを持つマイクロプロセッサの様なシステム LSI に適している。

2.2 準備

提案アルゴリズムを説明するために、以下の表記を用いる。

ドメイン: チップを 2 つ以上の区域に分け、それぞれの区域をドメインと呼ぶ。

ドメインサイズ: ドメインの縦の長さと、横の長さ。

電流密度変動係数: ドメイン内のインスタンスの電流値の平均と標準偏差を計算する。標準偏差を平均で割ったものが、電流密度変動係数。

正確な電圧ドロップ解析、基板ノイズの解析、チップがノイズを受けても正しく動作するかどうかの解析を行うには、以下の 4 つのモデル化が重要な項目である。

1） ノイズ（電圧ドロップ）の発生を正しくモデル化

2） 電圧ドロップによる遅延変動を正しくモデル化

3） 伝播経路を正しくモデル化

4） ノイズを受けた場合のアナログマクロの動作を正しくモデル化

ノイズの発生を正しくモデル化するには、ボード、パッケージ、チップの電源 /GND ネットワークの LRC（インダクタンス、抵抗、容量）を正しく抽出すること、各インスタンスの動作タイミングを正しく求め、その動作タイミングで電流変化を発生させることが重要である。それらをもとに解析を行うことによって、ノイズの発生（電圧ドロップ）を求めることができる。電圧ドロップは、現在市販されている CAD ツールを用い、求めることができる。しかし非常に解析に時間が必要である。

電圧ドロップによる遅延変動を正しくモデル化するには、各インスタンス毎に、電圧変動、負荷変動に依存した遅延変動を記載したライブラリを作成する必要がある。このライブラリは、市販 CAD ツールで作成することができる。電圧ドロップ結果とこのライブラリを用いてタイミング解析（STA）を行うことによって、デジタル回路が正しく動作するかどうかを判断することができる。

伝播経路を正しくモデル化するには、基板ネットワークの LRC（インダクタンス、抵抗、容量）を正しく抽出することが重要である。基板ネットワークの LRC 抽出は市販の CAD ツールで行うことができる。しかし、適用できる回路規模は小さく、リングオシレータ等の小規模回路にのみ適用可能である。

ノイズを受けた場合のアナログマクロの動作を正しくモデル化するには、ノイズ発生モデルと伝播経路のモデルをアナログマクロの回路ネットリストに接続をし、回路シミュレーションを行うことによって解析することができる。

正確な電圧ドロップ解析、基板ノイズの解析、チップがノイズを受けても正しく動作するかどうかの解析は、現在実用化されている方法では、処理時間、メモリの観点から、大規模回路に適用をすることは難しい [2]。そこで、大規模回路に於いても短時間に、精度良く解析が行える方法を提案する。

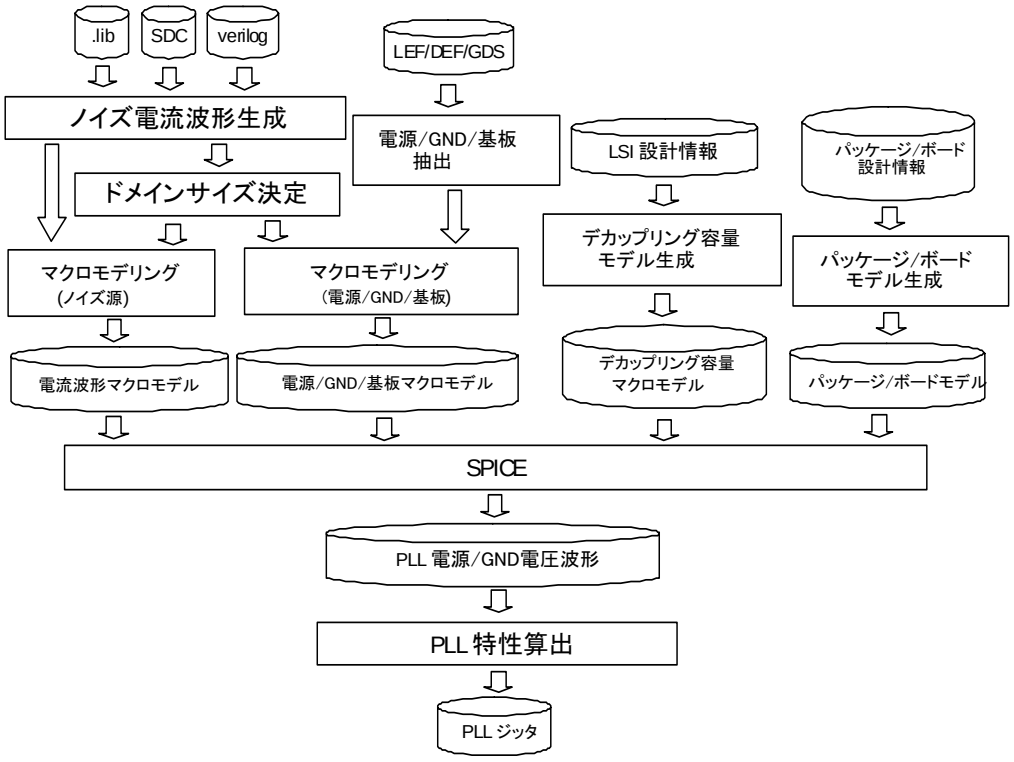

Figure 2.1: 提案の基板ノイズ解析フロー

2.3 解析フロー

図2.1に提案アルゴリズムのフローを示す。

STEP 1: ドメインサイズ決定　各インスタンスの電流波形を、 SDC ファイル [10] もしくは動作率を元に、消費電力ライブラリ (.lib) [10]とネット接続を元に、作成する。次に、 マクロモデルのドメインサイズが、チップの電流密度変動係数を元に、決定される。

STEP 2: ノイズ源モデルの作成　各マクロモデルの電流波形（ノイズ源モデル）は、個々のインスタンスの電流波形を足す事によって作成される。ノイズ源モデルは、 テストチップの測定結果からも作成することができる。

STEP 3: 電源 /GND/ 基板モデルの作成　電源 /GND/ 基板の RC (抵抗と容量) モデルが抽出される。次に、 RC モデルがマクロモデルのドメインサイズを元に、メッシュネットワークに縮退される。 RC モデルの抽出、縮退は、既存の CAD ツールを用いて行う。

チップに於いては、インダクタンスの影響が十分小さい [11]ので、電源 /GND/ 基板ネットワークのインダクタンスは抽出しない。

STEP 4: デカップリング容量モデルの作成　デカップリング容量は、それぞれのプリミティブブロック、デカップリングブロックから計算される。各マクロモデル単位でのデカップリング容量は、ドメインに含まれるインスタンスのデカップリング容量を足す事によって生成される。

STEP 5: パッケージと **PWB** モデルの作成　パッケージとプリント基板の電源 /GND ネットワークは、ランプド RLC モデルとしてモデル化される。これらのモデルは、電磁界解析ツールで解析され、モデル化される。

STEP 6: SPICE ネットリストの生成　STEP 2 で生成されたノイズ源モデルと STEP 4 で生成されたデカップリング容量モデルは、 STEP 3 で生成された電源 /GND/ 基板モデルに接続され、チップモデルが作成される。 STEP 5 で生成された、パッケージ、プリント基板の電源 /GND モデルは、チップモデルに接続され、 SPICE ネットリストが作成される。

STEP 7: PLL の電源 **/GND** 電圧波形の算出　PLL の電源 /GND 電圧波形を算出するために、回路シミュレーション (トランジェント解析) が実行される。

STEP 8: PLL 特性の算出　PLL の回路シミュレーションが、 PLL ジッタを算出するために実行される。

ここで述べたフローは、一般的なライブラリを用いているので、いろいろな種類のチップに適用可能である。

2.4 大規模チップのためのマクロモデリングによるノイズ解析アルゴリズム

大規模チップの基板ノイズを解析するために、即ち数百億のトランジスタを持つチップを、実用的な時間で解析するために、マクロモデリングアルゴリズムを提案する (図 2.2)。

提案アルゴリズムは、チップを複数のドメインに分割する。各ドメインの回路、電源 /GND/ 基板ネットワークは縮退され、マクロモデル化される。基板ノイズ解析は、作成されたマクロモデルを結合し、実行される。

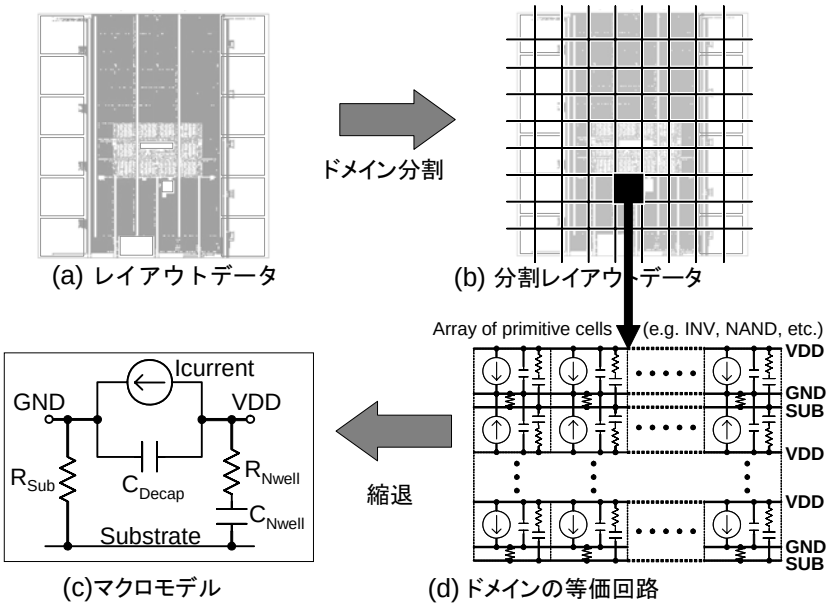

Figure 2.2: 基板ノイズ解析のためのマクロモデリングアルゴリズム

図 2.2 にマクロモデルの例を示す。基板ノイズの伝播経路は、 3 つのタイプに分類できる。

(1) 抵抗接続の GND コンタクト経由

(2) 容量接続の電源コンタクト経由

(3) 容量接続のトランジスタ経由

一般的なデジタル回路動作では、 (3) は小さい [12]ので本論文では考えない。

図 2.2 に於いて、 Rsub は、 GND コンタクトの抵抗を表す。一方 RNWell は電源コンタクトの抵抗を表す。 CNwell はドメインの NWell の容量を表す。 CDecap はデカップリング容量、 Icurrent は電流源を表す。 1 つのドメインに対し、 1 つのマクロモデルを作成する。そのマクロモデルは、 1 つの Rsub、 1 つの RNWell、 1 つの CNwell、 1 つの CDecap、 1 つの Icurrent を持つ。

マクロモデルの作成手順を以下に示す。

1. チップは、ドメインサイズ決定アルゴリズムを用いて複数のドメインに分割される (図 2.2）。

2. それぞれのドメインでは、ドメイン内のすべてのインスタンスの電流値とデカップリング容量値が計算され (図 2.2）、ドメインの Icurrent 値と CDecap 値がそれらの値を足す事によって求められる。

3. 各ドメインの Rsub 値、 RNwell 値、 CNwell 値が、幾何学的処理を用いた基板ネットワーク縮退アルゴリズムを用いて求められる。

4. 各ドメインに対し、ドメインのマクロモデルは、 2)、 3) で求めた Icurrent 値、 CDecap 値、 Rsub 値、 RNwell 値、 CNwell 値を用いて作成される。

このマクロモデリングは、計算複雑度を大幅に縮退できるので、実用的な処理時間で大規模チップの基板ノイズ解析を可能にする。

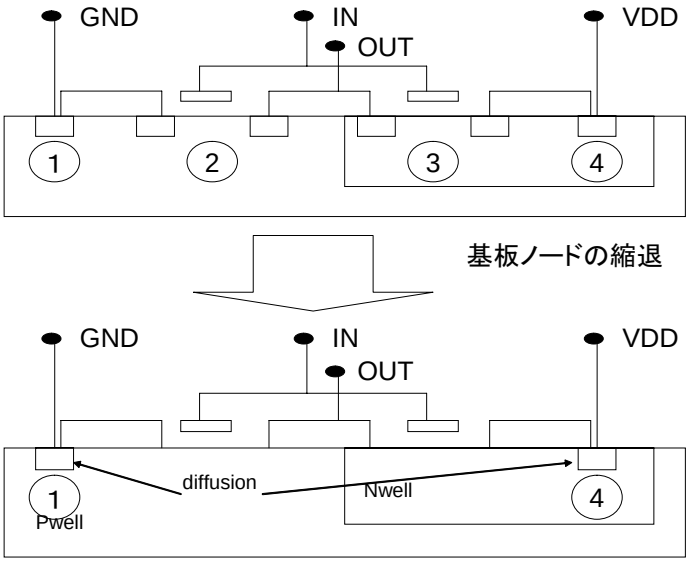

Figure 2.3: 基板ネットワーク縮退アルゴリズム

2.4.1 マクロモデリングの基本概念

基板抽出は、一般に 3-D finite difference method (FDM) もしくは boundary element method (BEM) で行われる。これらの方法は精度が高いが、扱える規模に制限がある。規模の問題を解決する為に、基板ネットワーク抽出方法 [13], [14]が提案されている。しかし、これらの方法は、以下の問題点がある。

1. ライブラリを作成する必要がある。

2. ライブラリが持つパタン以外には対応できない。

幾何学的処理を用いた基板ネットワーク縮退アルゴリズムを提案する。デジタル領域では、プリミティブブロックが並んで配置される。多くのコンタクトが、トランジスタのバックバイアスの電位を固定するために配置される。同じ拡散に多くのコンタクトが配置されるので、コンタクト抵抗は十分小さく、無視をして問題ない。拡散の下面と側面が基板の抵抗を決めるキーとなる。基板ネットワーク縮退アルゴリズムを以下に示す。

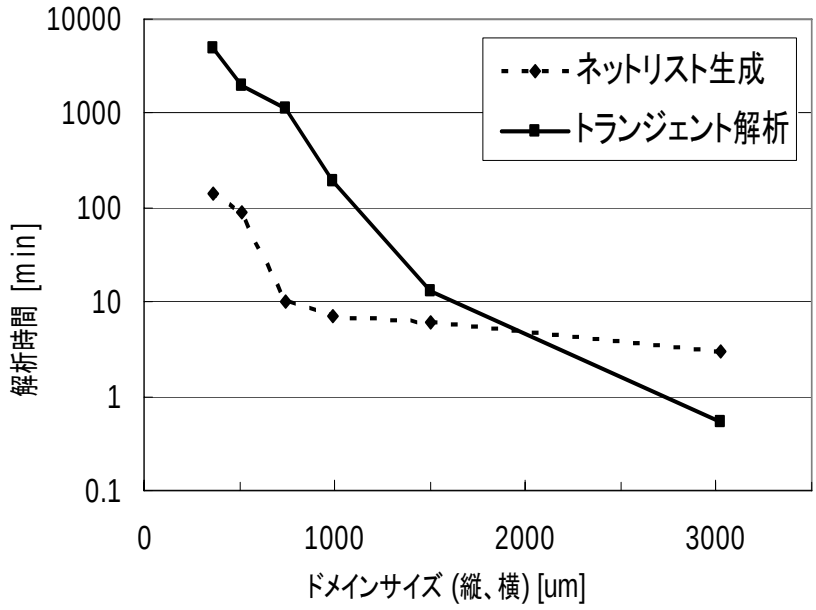

Figure 2.4: ドメインサイズと解析時間に関係

各ドメインに対して、

STEP 1: Pwell、 Nwell 内のすべての拡散の周囲長の合計を計算する。

STEP 2: STEP1 で求めた周囲長の合計と、同じ周囲長の拡散を作成する。

インバーター回路の基板モデルの例を図2.3に示す。トランジスタのバックゲートの接続は、提案アルゴリズムでは、考慮しない。基板ネットワーク縮退アルゴリズムを用いて、各ドメイン内のノードは、 1 個の Pwell、 Nwell にまで縮退される。ゆえに、ノードの数は、大幅に縮小され、マクロモデルを用いた解析時間は、短縮される。提案アルゴリズムは、幾何学的処理によって図形の数を縮小し、ライブラリの準備を不要とする。よって、基板ネットリストの抽出時間は大幅に短縮される。ドメインの数は、提案アルゴリズムを使って容易に変更可能で、分割の自由度は増加する。

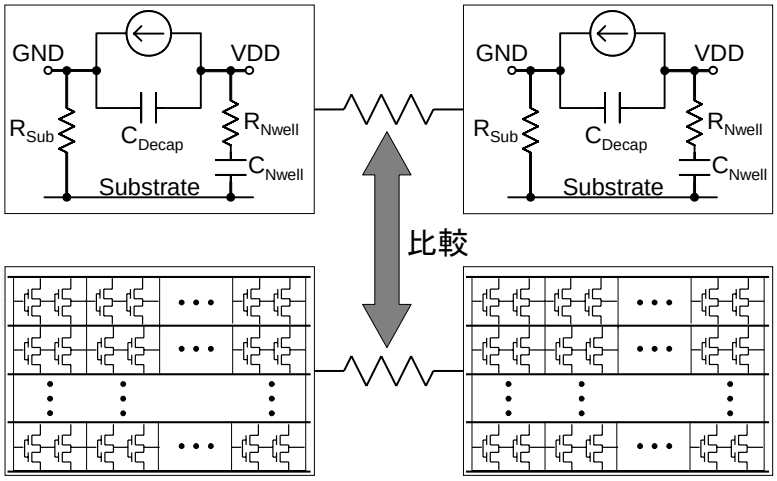

Figure 2.5: マクロモデルの基板抵抗誤差検証イメージ

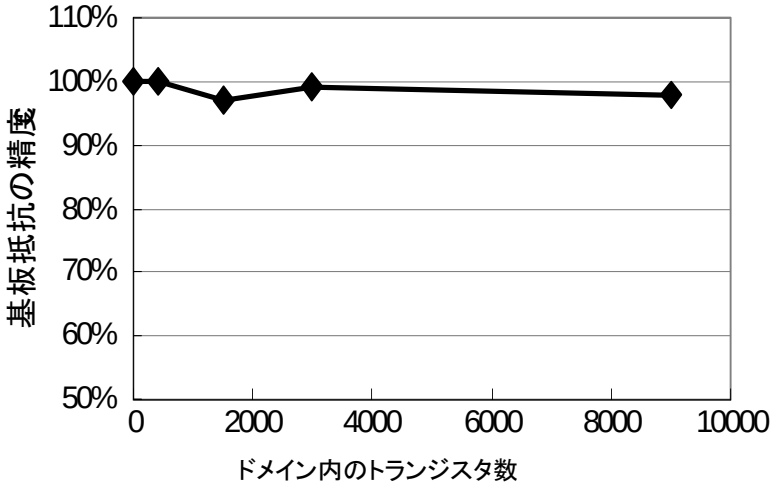

Figure 2.6: ドメイン内のトランジスタ数と基板抵抗の関係

2.4.2 ドメインサイズと解析時間の関係

解析時間を短縮するためには、ドメインをどのように分割することが効果的かを調査するために、 20mm x 21mm のチップデータを用いて、 CPU とドメインサイズの関係について調査を行った。結果を図2.4に示す。 横軸は、ドメインサイズを表し、縦軸は処理時間を表す。この図から、ある点を境に、ドメインサイズを小さくすると、指数関数的にネットリスト作成とトランジェント解析の処理時間が増加することが分かる。よって、ドメインサイズを減少させるには限界があることが分かる。ドメインサイズが大きいとき、縮退の比率が高くなる。もし、縮退の比率が高いと、解析精度が悪くなる。よって、解析時間と精度の間にはトレードオフの関係が成り立ち、このトレードオフを考える必要があることがわかる。

2.4.3 ドメインサイズと解析精度の関係

ドメインサイズと解析精度の関係を調査した。調査方法は、まず、 2 つのドメインに注目し、その 2 つのドメイン間の距離を 100um とした。次に、ドメインのサイズを変更し、基板抵抗値の変化を調べた。次に、 2 つのドメインそれぞれをマクロモデリングし、ドメインサイズを変更し、基板抵抗値の変化を調べた。そして、このマクロモデルを行ったものと、行わなかったもので基板抵抗値の比較を行った (図2.5)。結果を図2.6に示す。最大誤差は、ドメイ

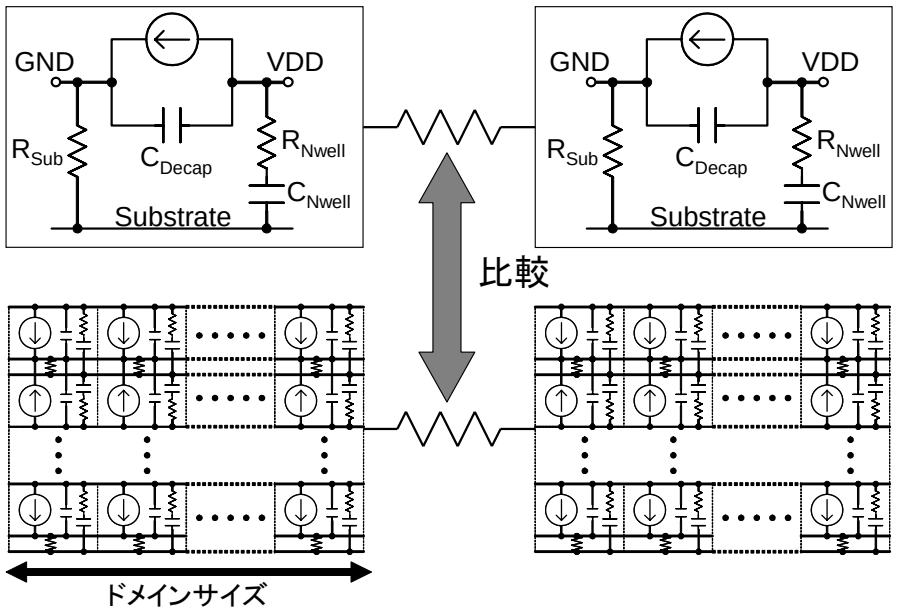

Figure 2.7: 階層に於けるマクロモデルの基板抵抗誤差検証イメージ

ンに含まれるトランジスタの数が 9000 までは 4% 以下であることがわかった。次に、新しく上記で用いたマクロモデルを用いて、 2 つのドメインを作成し (図2.7)、 精度を検証した。まず、 2 つの新たなドメインもマクロモデリングを行い、同様に基板抵抗の比較を行った。この結果から、基板抵抗の差は、ドメインサイズ 2000um までは、小さいことがわかった（図 2.8）。この結果から、ドメインサイズは基板抵抗にあまり影響を与えないことがわかった。

2.4.4 ドメインサイズ決定アルゴリズム

電流密度のばらつきが大きいデジタル回路の場合では、大きなドメインでは基板ノイズ見積もり精度が悪くなる。図2.9は、ドメインの電流密度変動係数と見積もり基板ノイズ誤差の関係を示したものである。この結果から、ドメインの電流密度変動係数と見積もり基板ノイズ誤差の間には線形の関係が存在する。例えば、基板ノイズの誤差を 15% 以下にするには（精度を 85% 以上にしたい場合）、電流密度変動係数を 7 以下にしなくてはならない。電流密度

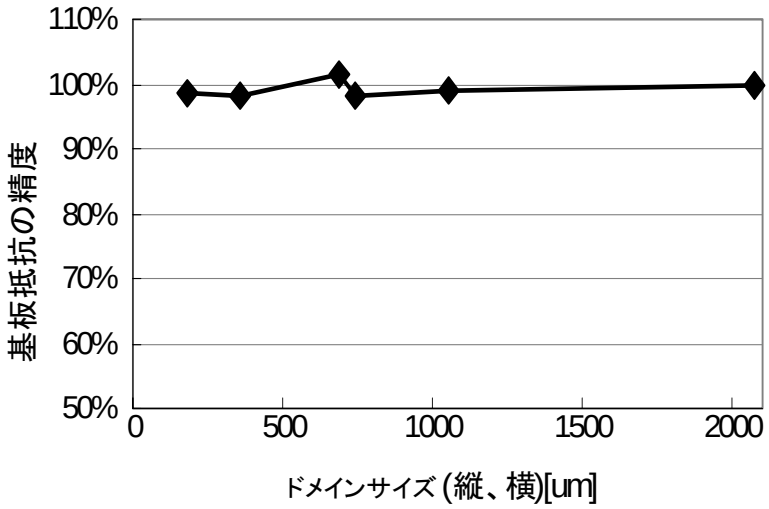

Figure 2.8: 階層に於けるドメインサイズと基板抵抗の関係

変動係数を小さくするには、電流密度変動係数が大きな値を持つドメインを、電流密度変動係数が要求値となるか、それ以下になるまで、繰り返し分割を行う (図2.10参照)。図2.11は、ドメインサイズと見積もり基板ノイズの精度の関係を示す。 繰り返しドメインを分割することによって、電流密度変動係数を減少させることが出来ることがわかる。分割は基板ノイズの見積もり精度を上げることになる。

以上のことから、マクロモデリングは以下の考えて行う。

1. 電流密度がほぼ同じドメインに対しは (図2.12), モデルは出来るだけ縮退する。
2. 電流密度が均一でないドメインに対しては (図2.13)、電流密度が均一になる様にドメインを分割し、モデル化する。
3. アナログマクロに対しては、コンタクトピッチと電流密度が均一ではないので、小さなドメインを適用する。

デジタル回路に適用する、ドメインサイズ決定アルゴリズムは、以下に示す 5 ステップから成る。

STEP 1: 各インスタンスの電流波形は、 SDC ファイルもしくは、動作率及び、電力ライブラリ、ネット接続情報を元に計算される。

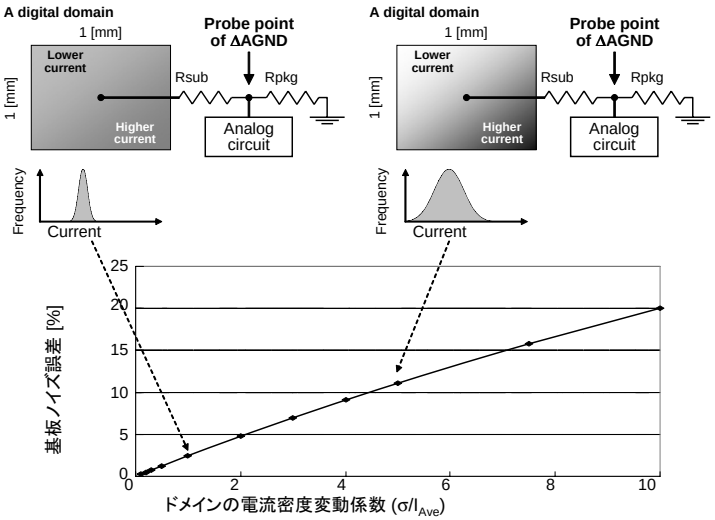

Figure 2.9: 電流密度変動係数と基板ノイズ誤差の関係

STEP 2: 目標の誤差から、電流密度変動係数の要求値を計算する。

STEP 3: チップのデジタル回路を、特定の初期サイズに分割する。

STEP 4: 各ドメインに対し、ドメイン内の各インスタンスの電流波形を用いて、電流密度変動係数を算出する。

STEP 5: 各ドメインに対し、もし、ドメインの電流密度変動係数が、要求値より小さい場合、ドメインの左下、右上座標を出力する。そうでない場合は、ドメインを分割し、 STEP4 へ。

2.5 計算機実験結果

提案のドメインサイズ決定アルゴリズムの精度を検証する目的で、小さなデータを用いてテストした。テストは、 2.6GHz、 4GB メモリのリナックスワークステーションでおこなった。プリミティブブロックレベルマクロモデリングアルゴリズムもプログラム化した。テーブル 1 に実験結果を示す。 #node と #domain は、プリミティブブロックの数と分割後のドメインの数を示す。 Target err と err は、目標誤差と従来のプリミティブブロックレベルマクロモ

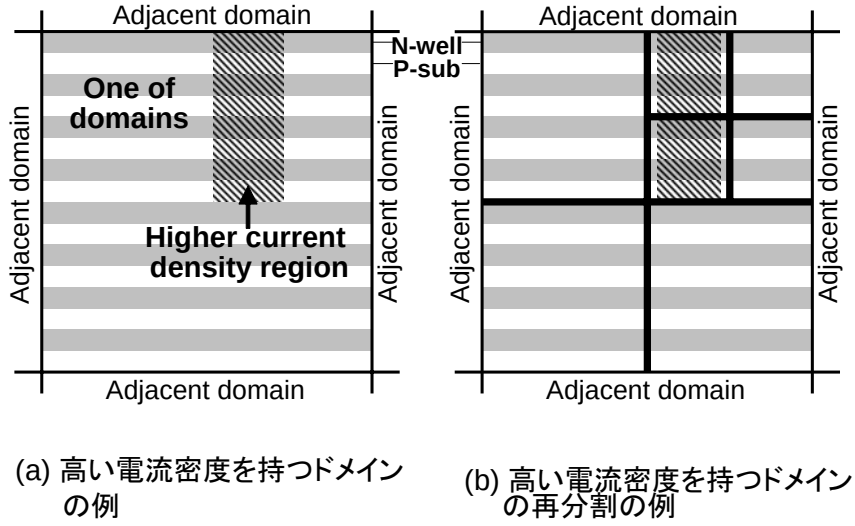

Figure 2.10: ドメイン分割と、ドメインの再分割

デリングアルゴリズム、提案アルゴリズムを適用した場合の誤差を示す。 CPU は、ドメインサイズ決定の処理時間、マクロモデリングの処理時間、基板ネットリスト作成の処理時間、トランジェント解析の処理時間の合計を示す。 Speed up は、プリミティブブロックレベルマクロモデリングアルゴリズムを用いた従来アルゴリズムと提案アルゴリズムの解析時間の割合を示す。図2.14は、データ３のドメインサイズ決定アルゴリズムの実験結果である。データ３のメモリ使用量は、 911MB であった。計算機実験結果から、目標誤差と誤差の間の差は、 0.3% より小さかった。プリミティブブロックレベルマクロモデリングアルゴリズムを用いた従来アルゴリズムと比べ、提案アルゴリズムは、 12-40 倍高速であった。処理時間の高速化効果は、多くの数のノードを持つデータでより大きくなる。これらの結果から、提案のドメインサイズ決定アルゴリズムは、とても有効なアルゴリズムであることを確認した。

我々は、提案のドメインサイズ決定アルゴリズムをスーパーコンピュータ用プロセッサに適用した。図 2.15 は、プロセッサの電流密度を表す。精度を確保するために、繰り返し、大きな電流分散を持つデジタル領域を各ドメインの電流密度変動係数 $(\sigma/Iave)$ が平均電流密度（Iave）の 3 倍になるまで分割した。テーブル２は、電流密度変動係数が要求誤差を満足しているドメイン、満足していないドメインの数を示す。まず、ドメインサイズを 2688 um と

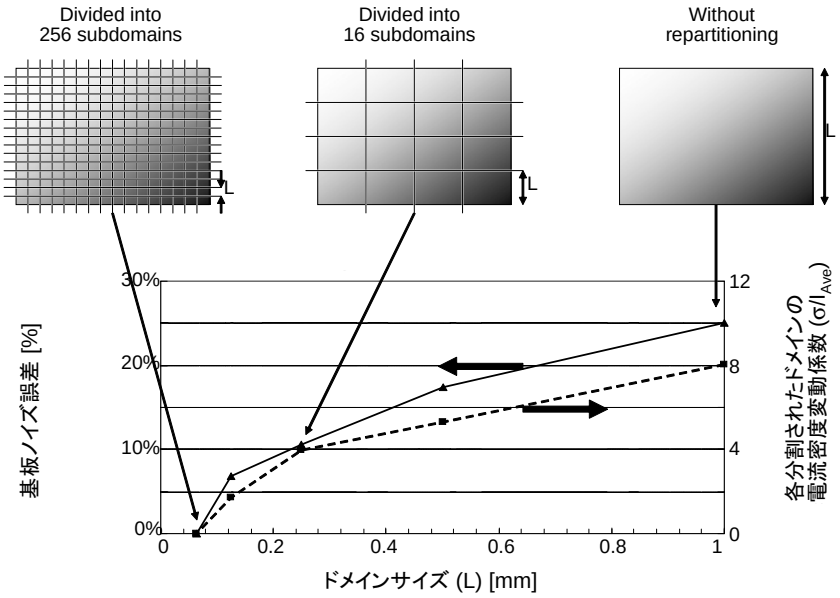

Figure 2.11: 基板ノイズ誤差とドメインサイズの関係

た。最初の分割で、 40 ドメインが要求の電流密度変動係数 ($\sigma/Iave \Leftarrow 3$) を満足し、 10 ドメインが満足することが出来なかった。 10 ドメインは、 40 のサブドメインに分割された。 2 ドメインは要求を満足できたが、 13 個は満足することが出来なかった。要求の電流密度変動係数を満足できないドメインは、再度分割され、最終的には、ドメインサイズ 168 um で要求を満足することができた。 図2.16は、各分割ステップでの電流密度変動係数 ($\sigma/Iave$) の変化を示す。これは、電流密度の分散が大きいドメインが、サブドメインに分割される様子を示す。結果、プロセッサチップは 2688um のサイズを持つ 40 ドメイン、 1344um のサイズを持つ 2 ドメイン、 672um のサイズを持つ 5 ドメイン、 336um サイズを持つ 15 ドメイン、 168um サイズを持つ 12 ドメインを含む 74 ドメインに分割された。各ドメインのネットリストを縮退し、マクロモデルを作成することによって、解析速度と精度の両方を得ることが出来た。

提案のマクロモデリングアルゴリズムの精度を検証するために、テストチップを作成し、電源ノイズが原因の PLL の周期ジッタを測定した。テストチップはスパーコンピュータのマイクロチップと同じサイズで同数のトランジスタを搭載している。 PLL 回路はチップの中央に置かれ、周りを囲むようにデジタルノイズ源であるコアのデジタル回路を配置した。 PLL 回路の電源 /GND ネットワークはコアの電源 /GND ネットワークとメタル配線を経由し

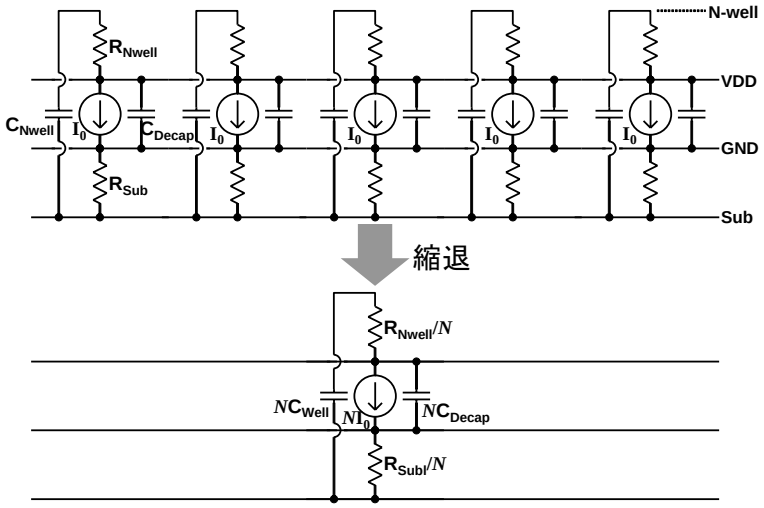

Figure 2.12: モデル作成ポリシイ: 電流密度がほぼ同じ

たノイズ伝播を防止する目的で分離された。よって、ノイズの伝播経路は Si 基板のみである。コアの論理回路は、可変クロックゲーティング機能を有し、ゲーティッドサイクルを変化させることが出来る。ストリーミングデータはクロックの立ち上がりエッジで変化する。実験では、コア回路のゲーティッドサイクルをパラメータとし、 31.25 MHz から 250 MHz まで変化させた。 PLL のジッタ測定は、オシロスコープ (Infinium 54855A by Agilent Technology) で行った。各ゲーティッドサイクルで電源ノイズのピークを測定した。結果を図 2.17に示す。回路シミュレーションにおいては、この電源ノイズは電圧源としてモデル化されコア回路の電源、GND 間に挿入された。 GND の揺れは GND コンタクトを介して基板に挿入されると仮定した。

1mm x 1mm のサイズにテストチップをドメイン分割し、マクロモデリングした。ノイズ解析にかかった処理時間は、 1.6GHz CPU 、 4GB メモリの Sun リナックスワークステーションを用いて、おおよそ 24 時間であった。図2.18で、回路シミュレーションにおけるクロックサイクルに於ける周期ジッタの割合を実測と比較した。この結果から、回路シミュレーションの結果は、実測と比較して、周期ジッタにおいて、誤差は、すべてのクロックゲーティング

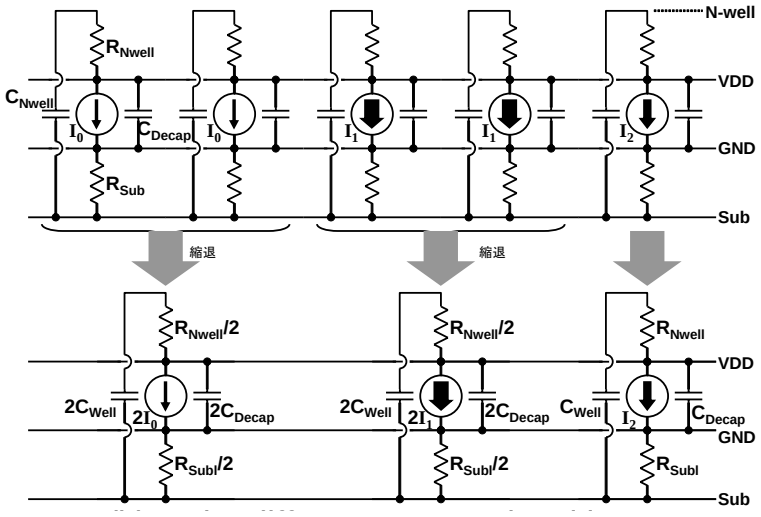

Figure 2.13: モデル作成ポリシイ: 電流密度が均一ではない

で 15% より小さい (i.e., 提案アルゴリズムの精度は、 85% 以上) 事が分かった。この精度は、スーパーコンピーターで用いるプロセッサのためには十分な精度である (図2.18)。

2.6 まとめ

大規模チップに対して、高速なマクロモデリングによるノイズ解析アルゴリズムを提案した。まず、チップは、複数のドメインに分割され、それぞれのドメインの回路は 1 つのマクロモデルに縮退される。精度を保持するために、回路タイプもしくは電流密度変動係数によってドメインサイズを適時変化させる。この方法を使うと、精度と計算複雑度の低減両方を実現することが出来る。

提案アルゴリズムの精度を検証するために、テストチップを作成し、 PLL 回路の周期ジッタを測定した。そして、測定結果と回路シミュレーション結果を周期ジッタで比較した。比較の結果、実測と回路シミュレーション結果の差は、すべてのクロックゲーティッドにおいて、 15% 以内であることが確認できた。また、クロックジッタ解析を含んだ解析時間は、おおよそ 24 時間であった。

Table 2.1: 計算機実験結果

	#node	#domain	CPU(s)	Speedup	target err	err
data1	257	17	80.7	12.2x	3%	2.82%
data1	257	5	29.08	23.5x	5%	4.73%
data2	1281	81	80.4	14.4x	3%	2.97%
data2	1281	21	29.1	39.5x	5%	4.8%
data3	6167078	254	89116	—	5%	—

Table 2.2: ドメインサイズ決定過程

	Domain size (side length) [um]				
	2688	1344	672	336	168
OK ($\sigma \leq 3I_{ave}$)	40	2	5	15	12
NG ($\sigma > 3I_{ave}$)	10	13	11	4	0
No count (PLL, SerDes, etc.)	6	25	34	25	4

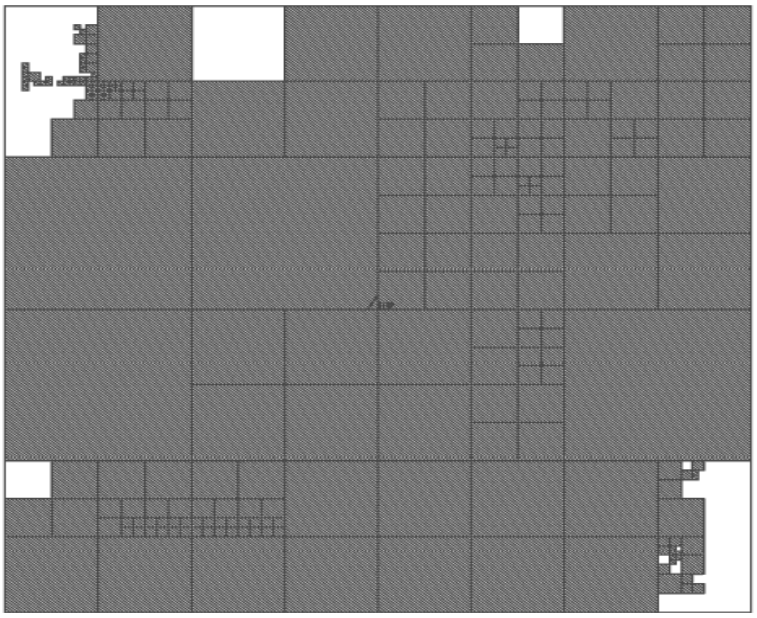

Figure 2.14: データ３に対するドメインサイズ決定結果

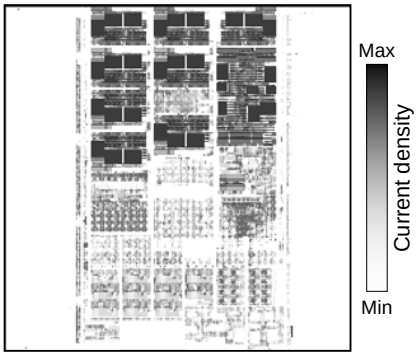

Figure 2.15: 実データに於ける電流密度

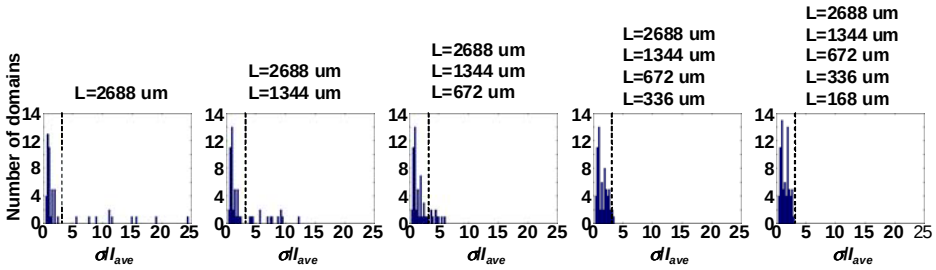

Figure 2.16: ドメインサイズ決定過程における σ / Iave の 改善

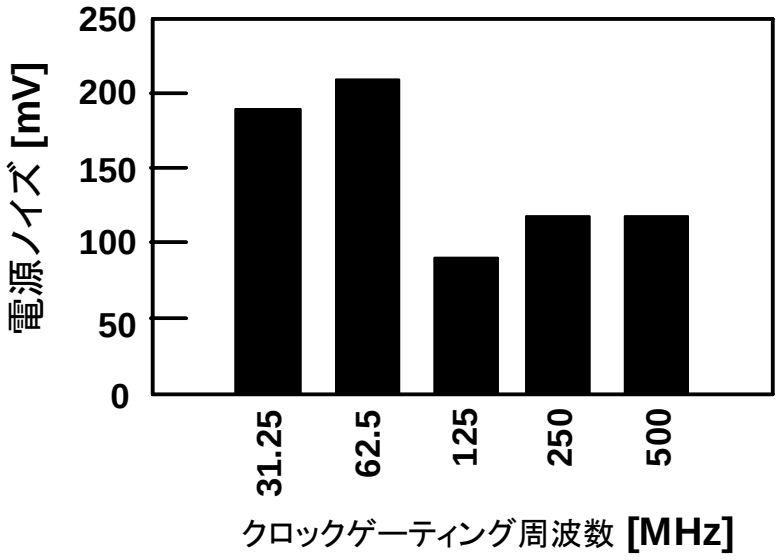

Figure 2.17: クロックゲーティング周期の電圧ドロップ依存

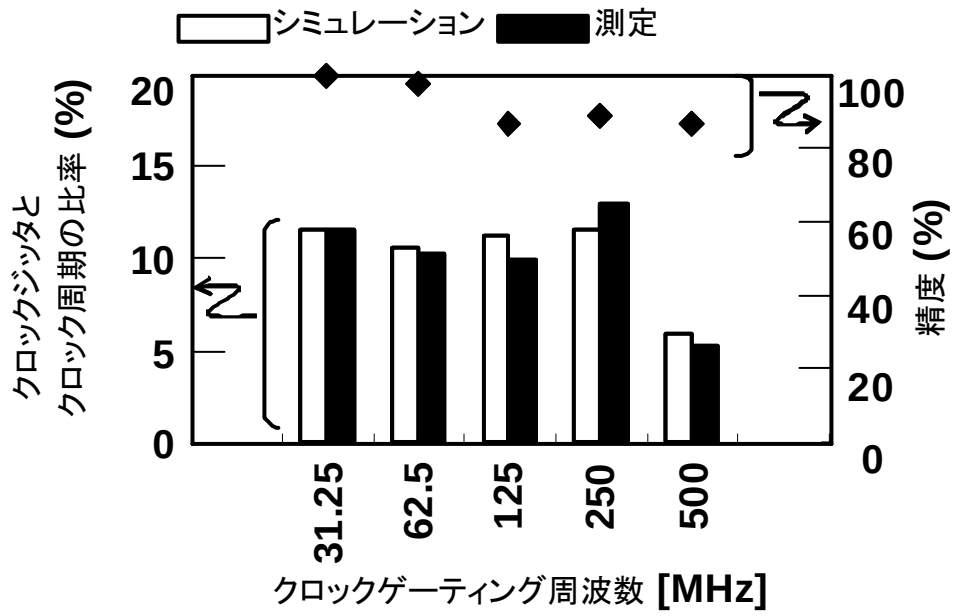

Figure 2.18: 周期ジッタの実測、回路シミュレーション比較

Chapter 3

電源 /GND ネットワーク最適化アルゴリズム

3.1 概要

近年、電源 /GND ネットワークの配線はデザインに於ける配線リソースの 20% 以上を占めている。電源 /GND ネットワークの配線の面積の増加は、配線性とタイミングクロージャーに大きな影響を与えている。また、プロセス技術の進歩に伴い、多層配線が一般的に用いられる事となってきた。しかし、製造コストと製造 TAT の面から考えると、配線層数を減らすこと、すなわち、電源 /GND ネットワークの配線の総面積を減らすことは重要である。

加えて、プロセス技術の進歩に伴い、トランジスタの動作周波数は上がり、搭載される素子の数も増加した。この結果、ノイズは増加した。加えて、電源電圧は低下し、信号の論理振幅は小さくなった。この結果、電圧ドロップが信号に与える影響は、相対的に大きくなった。ゆえに、より電圧ドロップの問題は重要となってきた。

電圧ドロップを減らす最善の方法は、電源 /GND ネットワークの配線の抵抗を減少させることである。しかし抵抗を減少させることは、電源 /GND ネットワークの配線の面積を増加させることとなる。ゆえに、非線形計画法 [15], [16]を用いた 電源 /GND ネットワークの配線面積を最小化し、電圧ドロップ制約を満足するアルゴリズムが提案されている。アルゴリズムは、逐次線形計画法（Successive Linear Programming、 SLC) を用いる。非線形計画問題を解くための多くのプログラム [17], [18] が提案されており、これらはアクセス可能である。しかしこれらのアルゴリズムは、実際の SoC のような大規模回路に適用することは難しく、緩和のために精度が悪化する。 [16] に於いては、緩和によって、解の探索経路が制限され、解の精度が劣化した。そこで、欲張り最適化アルゴリズム（greedy optimization algorithms） [19], [20]が提案されている。アルゴリズムは、各繰り返しで電圧ドロップ違反を探すために回路シミュレーションを行う。電源 /GND ネットワークの配線の抵抗は、違反を減らすために変更され、電圧ドロップ違反をチェックするために再度回路シミュレーションが行われる。この処理は、繰り返され、解が得られる。詳細アルゴリズム [20]を以下に示す。

STEP1 電源 /GND ネットワークの配線抵抗値に初期値として、最も大きな値にセットする。

STEP2 回路シミュレーションを実行し、ノードの電圧を計算する。

STEP3 ノードの電圧は電圧ドロップ制約と比較され、もし、制約を満足していなかったら、枝の抵抗は均一に小さくし、 STEP2 へ。もし満足していたら、処理を終了する。

このアルゴリズムの 1 つの欠点は、均一に電源 /GND ネットワークの配線の抵抗を変更するので、電源 /GND ネットワークの配線面積が大きくなることである。そこで、独立に電源 /GND ネットワークの配線の抵抗を変更することによって配線面積の最小化をおこなうアルゴリズムを提案する。提案アルゴリズムは、電圧ドロップに導く原因の配線を見つけ、その配線の幅を変更する。提案アルゴリズムはネットワークフローアルゴリズム [21]を用いている。提案アルゴリズムは、配線の総面積が最小になるように、配線幅最小から処理を始める。次に、電圧ドロップに導く最も影響のある枝、クリティカル枝を捜し、その枝に対応する配線の幅のみを変更する。配線幅は、ノードの電圧ドロップ違反を解決できる幅に変更されるので、面積増加を最小限に抑えることができる。ゆえに、提案アルゴリズムは電圧ドロップ制約を満足し、電源 /GND ネットワークの配線の面積を最小化する。提案アルゴリズムは, 電圧ドロップに導くクリティカル枝を見つけ、改善するので、有効なアルゴリズムである。

3.2 フロアプランフェーズにおける電源 /GND ネットワーク設計

3.2.1 SOC 設計フローにおける電源 /GND ネットワーク設計

実際の設計に於いては、フロアプランが完了した時点で、ボード、パッケージの設計が開始される。ボード、パッケージ設計を行うには、ターゲットインピーダンス設計のために、チップモデル [22] が必要である。チップモデル作成のためには、電源 /GND ネットワークの配線設計が完了している必要がある。ゆえに初期配置、配線のフェーズではなく、フロアプランのフェーズで電源 /GND ネットワーク設計を行う。

SoC の設計フローを図3.1に示す。 マクロブロックの位置、 PAD の位置、消費電力、動作周波数は論理合成とフロアプラニングで決定される。即ち、マクロの位置と電流は決定される。この条件で、フロアプラニングフェーズで、電圧ドロップを満足し、電源 /GND ネットワークの配線の総面積を最小化する電源 /GND ネットワーク設計問題を解く。

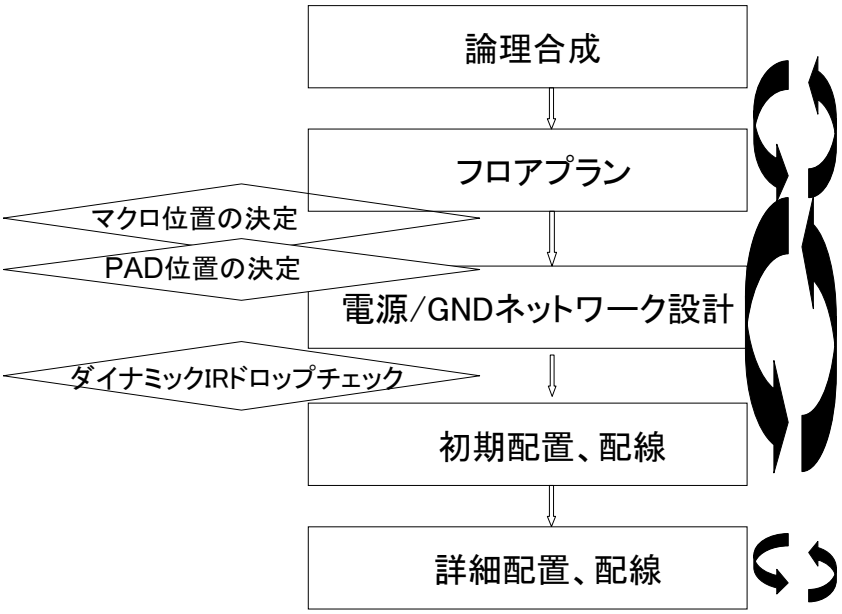

Figure 3.1: SoC 設計のデザインフロー

3.2.2 ノード電流の決定

フロアプラニングフェーズにおいて、多くの場合マクロ内の電流分布は決定されていない。そこで、マクロブロックの電流情報から、ノードの電流を決定する方法 [15, 23, 24] をここで述べる。もし、長方形要素モデルが使われたなら、電源 /GND ネットワークモデルは十分な精度があることが知られている。そこで、それぞれの長方形が長方形に対応するノードを含む様に、チップは長方形の集合に分割される。図3.2にチップを分割した結果の例を示す。点線は、長方形境界を表し、実線は、電源グリッドを表す。この例では、 4 つのマクロブロック I, Ⅱ, Ⅲ, Ⅳ が配置されている。マクロブロックの面積比によってマクロ電流は分割され、長方形ドメインの電流とされる。長方形ドメインの電流は、ノードの電流としてセットされる。

3.3 電源 /GND ネットワーク設計問題の準備

3.3.1 電源 /GND ネットワーク回路シミュレーション

本章では、電源 /GND ネットワーク回路シミュレーションについて説明する。チップの電源 /GND ネットワークは、固定電圧と時間変化電流源を持つ RLC ネットワークとしてモデル化される。この問題は、典型的な方式 Modified Nodal Analysis (MNA) [25] 方式で解くことができる。

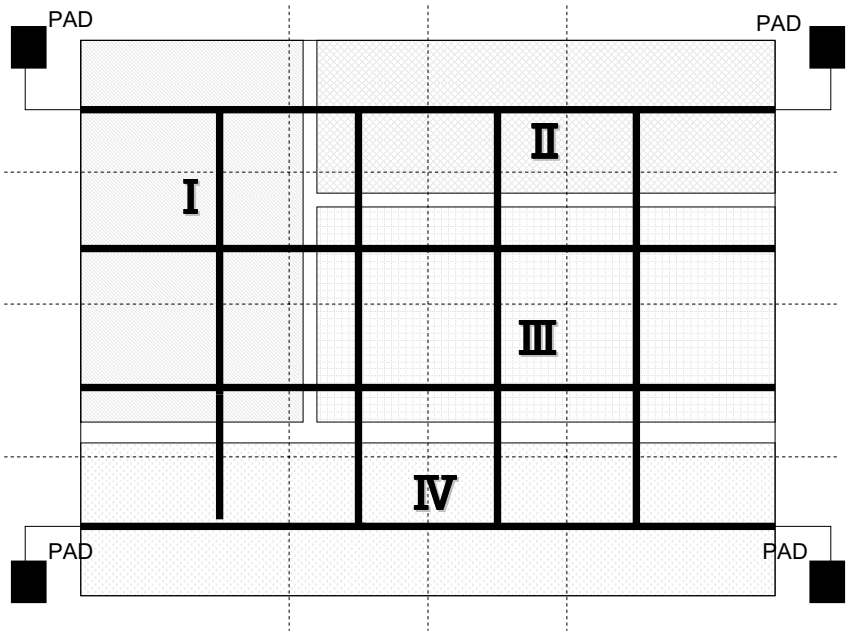

Figure 3.2: フロアプランフェーズに於けるノード電流の決定

$$G \times x(t) + C \times x'(t) = b(t) \tag{3.1}$$

ここで、G はコンダクタンスマトリックス、C は、アドミッタンスマトリックスを示す。$x(t)$ は電圧の時間変化ベクターで、$b(t)$ は、電流源の時間変化ベクターを示す。MNA システム (式 3.1) は、以下の線形代数システムによって効果的に解く事ができる。

$$(G + C/h) \times x(t) = b(t) + C/h \times x(t-h) \tag{3.2}$$

この式を、固定ステップ h で 後退 Euler (BE) 法 で解く。このアルゴリズムは回路シミュレーションのトランジェン解析で用いられており、効果的に解く事ができることが知られている。通常チップの電源 /GND ネットワーク回路シミュレーションにおいて、インダクタンスは無視される。これは、チップのインダクタンスの値が小さく、解析結果への影響が小さいためである。ゆえに、本論文ではチップのインダクタンスを考慮しない。我々は、MNA 方式を用いた電源 /GND ネットワーク設計アルゴリズムについて提案する。

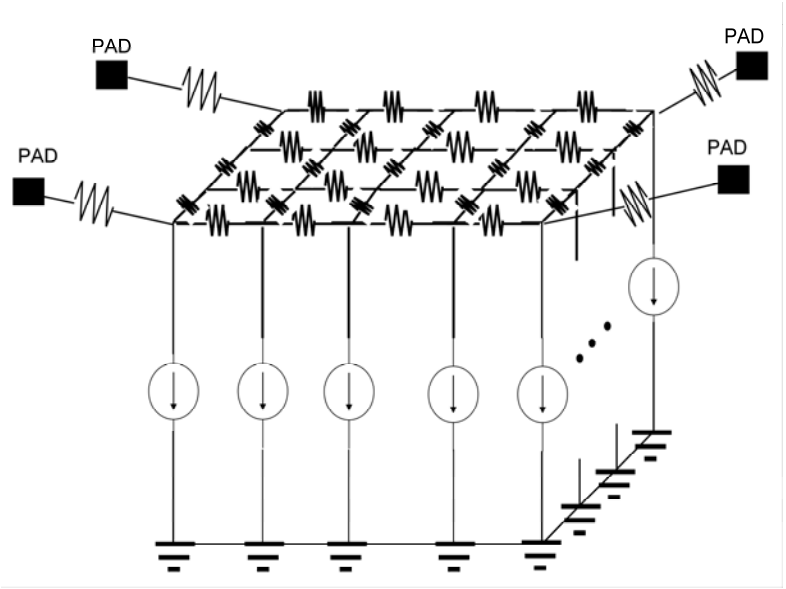

Figure 3.3: 電源の抵抗メッシュモデル

3.3.2 電源 /GND ネットワーク回路シミュレーション高速化アルゴリズム

数億ノードの電源 /GND ネットワークを実用的な処理時間で解く事は難しい。そこで、電源 /GND ネットワーク設計問題を、いくつかの電源 /GND ネットワークに分割し、そのサブ電源 /GND ネットワーク設計問題を個々に解くことによって、高速化を行う。分割された個々の電源 /GND ネットワークは、多ポート線形要素 [20]としてモデル化できる。

$$I = A \times V + S, \tag{3.3}$$

ここで

A : アドミッタンスマトリックスのポート

V : ポートでの、電圧ベクトル

I : ローカルグリッドとグローバルグリッド間のインターフェースを通過する電流ベクトル

S : それぞれのポートとリファレンスノード間を通過する電流ベクトル

電源 /GND ネットワーク回路シミュレーション高速化アルゴリズムの詳細は [19, 20]に述べられている。

3.3.3 フロー概要

提案アルゴリズムのアウトラインをここで示す。提案アルゴリズムでは、初期解として、配線幅を最小とする。配線幅を増やす処理のみで、電圧ドロップの改善を行い、電圧ドロップ制約を満足する解を求める。提案アルゴリズムを以下に示す。

初期解として、配線面積を最小とする。
電源 /GND ネットワーク回路シミュレーションを実行する。
While（電圧ドロップ制約を満足していない解析時間ある。）
{
電圧ドロップ制約を満足していない解析時間に対して、電圧ドロップに
導く最も影響の大きい枝、即ちクリティカル枝を捜し、その枝に相当する
配線の幅のみ、電圧ドロップ制約を満足する値に変更する。
電源 /GND ネットワーク回路シミュレーションを実行する。
}

これ以降、電圧ドロップ制約を満足しない解析時間に対する提案の電圧ドロップ改善アルゴリズムの詳細を説明する。

3.4 電源 /GND ネットワーク設計問題の定式化

図3.3は、電源ネットワークをモデル化した抵抗メッシュモデルを示す。ここで、それぞれの電源グリッドの抵抗は、多層の配線の抵抗を合成したものである。説明を簡単にする目的で、電源グリッドは、同じ間隔で生成するものとする。電源ネットワークの抵抗メッシュモデルは、電圧ドロップを満足しない解析時間の回路シミュレーションの結果から作成される。枝の抵抗は、ノードの電圧と枝の電流値から計算される。ノードの電流は、電源ネットワークのノードから GND ネットワークのノードへ流れる電流である。パッケージのインダクタンスは抵抗として表現される。チップの容量は考えない。 GND ネットワークは、同様にモデル化される。ここでは、電源ネットワーク設計問題を解くアルゴリズムのみ述べる。なぜなら、 GND ネットワーク設計問題は、電源ネットーワーク問題を解くアルゴリズムと同じアルゴリズムで解く事ができるからである。電源ネットワークグラフ $= (N_u, E_b)$ は、電源ネットワークの抵抗メッシュモデルとして表現される。ここで用いる記号を以下で定義する。

N_d: 電源ネットワークのノードの集合

N_s: PAD のノードの集合

$N_u = N_d \cup N_s$

$n = |N_u|$

E_p: N_d のノードと N_s のノードを結ぶ枝の集合

E_g: N_d のノードのペアを結ぶ枝の集合

$E_b = E_p \cup E_g$

$c_{i,j}$: 枝 $\{i, j\}$, $i, j \in N_u$ の電流

v_i: ノード $i, i \in N_u$ の電圧

$g_{i,j}$: 枝 $\{i, j\}$, $i, j \in N_u$ のアドミッタンス

$r_{i,j}$: 枝 $\{i, j\}$, $i, j \in N_u$ の抵抗

i_i: ノード i、$i \in N_u$ の電流源

3.4.1 電圧ドロップ制約

デジタル論理が正しく動作をするには、電圧ドロップ制約を満足しなくてはならない。この値は、STA （Static Timing Analysis) のシステムによって決定されるデザインマージンから決定される。. V_{min} と V_{max} は電源ネットワークの最小電圧と最大電圧として定義される。ゆえに、電圧ドロップ制約は以下のように表現することができる。

$$V_{min} \leqq v_i \leqq V_{max} \tag{3.4}$$

3.4.2 回路制約

ノードの電圧の集合は、電圧のベクトル $V = \{v_1, v_2, , v_n\}^T$ として表現することができる。同様に、ノードの電流の集合は、電流のベクトル $I = \{i_1, i_2, , i_n\}^T$ として表現することができる。キルヒホッフの電流則より、回路方程式を作成することができる [25]。回路制約は以下で表現することができる。

$$A \times V = I, \tag{3.5}$$

ここでA は $n \times n$ アドミッタンスマトリックスである。回路制約作成の詳細は [25, 26]に記載されている。

3.4.3 デザインルール制約

LSI デザインルールにおいて、配線の最小幅、最大幅、配線間隔、ビア最小幅、ビア最大幅、ビア間隔は定義されている。これらのデザインルールを守って設計することが必要である。加えて、配線面積にも制約がある。ゆえに、アドミッタンスは最大と最小の両方の制約を持ち、デザインルール制約として要求される。ネットワーク配線の最大、最小アドミッタンスを、G_{min} 、 G_{max} として定義する。よって、デザインルール制約は以下のように定義できる。

$$G_{min} \leqq g_{i,j} \leqq G_{max} \tag{3.6}$$

3.4.4 目的関数

電源ネットワークの配線面積を最小化することは、信号配線の配線性を高め、タイミングクロージャーを早め、製造コストを最小化するのに有効である。ゆえに、目的関数は、電源ネットワークの総配線面積を最小とすることである。$l_{i,j}$ と $w_{i,j}$ はそれぞれ、枝 {i, j} の横幅、縦幅を表す。シート抵抗を $\rho(\Omega$ / square) とする。電源ネットワークの総配線面積は以下の様に定義される。

$$Area = \sum_{i,j \in Nu} l_{i,j} \times w_{i,j} \tag{3.7}$$

また、枝 $\{i, j\}$ の抵抗は、以下の様に定義される。

$$r_{i,j} = \rho \times l_{i,j}/w_{i,j} \tag{3.8}$$

配線の長さは、チップサイズで決定される定数、即ち、$l_{i,j}$ 、であるので、配線の面積は、$w_{i,j}$ の和、即ち、すべての抵抗の最大和を最小化することにとって、減少させることができる。アドミッタンスは抵抗の逆数なので、配線面積は、すべてのアドミッタンスの和を減少させることによって、最小化できる。よって目的関数は以下で定義できる。

$$Obj : \min \sum_{i,j \in N_u} g_{i,j} \tag{3.9}$$

3.4.5 電源ネットワーク設計問題

以上の結果から、電源ネットワーク設計問題は、以下で定義できる。

$$\begin{aligned} Minimize \quad & \textstyle\sum_{i,j \in N_u} g_{i,j} \\ Subject \ \ to \quad & V_{min} \leqq v_i \leqq V_{max} \\ & G_{min} \leqq g_{i,j} \leqq G_{max} \\ & A \times V = I \end{aligned} \tag{3.10}$$

この問題を解くことによって、電圧ドロップ制約を満足し、電源ネットワークの配線の総面積を最小化する電源ネットワーク設計が実行される。

3.5 電源ネットワーク最適化アルゴリズム

3.5.1 提案アルゴリズムの概要

提案アルゴリズムは、電圧ドロップを満足しない解析時間の回路シミュレーション結果から計算された配線幅から始める。次に電圧ドロップ制約違反ノードの改善に最も有効な枝を見つけ、その枝に対応する配線の幅のみを変更する。ノードの電圧ドロップ制約違反を解く事ができる幅に配線を変更するので、面積の増加を最小限に抑えることができる。この処理をすべてのノードの違反がなくなるまで繰り返す。よって、配線面積を最小化でき（必ずしも大域的最小点ではない）、処理時間の増加を抑えることができ、制約を満足する解を得ることができる。

提案アルゴリズムのアウトラインを述べる。提案アルゴリズムは以下の処理を繰り返す。

STEP1 電源ネットワークの抵抗メッシュモデルを電圧ドロップ制約を満足しない解析時間の回路シミュレーション結果から作成する。

STEP2 抵抗メッシュモデルの回路シミュレーションを行い、ノードの電圧を計算する。

STEP3 電圧ドロップが最も大きいノードを見つける。

STEP4 電圧ドロップの最も大きいノードの電圧を決定しているパスを求める。

STEP5 パス中から最も電圧ドロップの大きいノードの電圧に影響のあるパスを見つける。

STEP6 パス上の枝で、電圧ドロップの最も大きいノードの電圧に最も影響のある枝を見つける。

STEP7 枝の幅を更新し、 STEP2 へ

以降では、アルゴリズムの各ステップに沿って、提案アルゴリズムの詳細を説明する。

3.5.2 ターゲットノードの抽出

抵抗メッシュモデルの回路シミュレーションを実行し、電圧ドロップ制約を満足しないノードを見つける。電圧ドロップ制約違反ノードを解消するために、配線面積の増加が最小になるように、配線の幅を変更する。最も電圧ドロップの大きいノードは、最も改善すべきノードであるので、そのノードに対し処理は行う。この処理は、制約違反ノードに対し行われる。すべての電圧ドロップ制約違反ノードの電圧改善の処理が行われ、総配線面積を最小化する解が求められる。

電圧を改善すべきノード、ターゲットノード選択を図3.4を使って説明する。図3.4で、ノードの横に記載されている数字はノードの電圧を表す。矢印は電流の流れを表す。図 3.4において、 0.8 は、最も電圧ドロップが大きいノードである。よってこのノードが選択される。

3.5.3 オーバーフローパスの抽出

ノードの電圧ドロップ違反を解消するために、改善できるパスを選択する必要がある。そこで、ノードの電圧を改善するパスを見つける方法について述べる。ノードの電圧は、 PAD から、そのノードまでのパスの抵抗と電流によって決まる。電流は、電圧の高いノードから低いノードへ流れる。電流の経路はパスである。よって、電圧の最も低いノードから電流経路を探索することによって、パスを見つける。オーバーフローパスを PAD から最も電圧の低いノードへのパスと定義する。

図 3.4で網掛け部分のパスは電流経路として選択された。最も電圧の低いノードの電圧を決めるパスである。図3.5は、抽出の結果を表す。図 3.5において、 $\{E, A, B, F, J, I, H, G\}$, $\{E, F, J, I, H, G\}$, $\{E, I, H, G\}$, $\{E, D, H, G\}$, $\{E, D, C, G\}$ は、オーバーフローパスである。

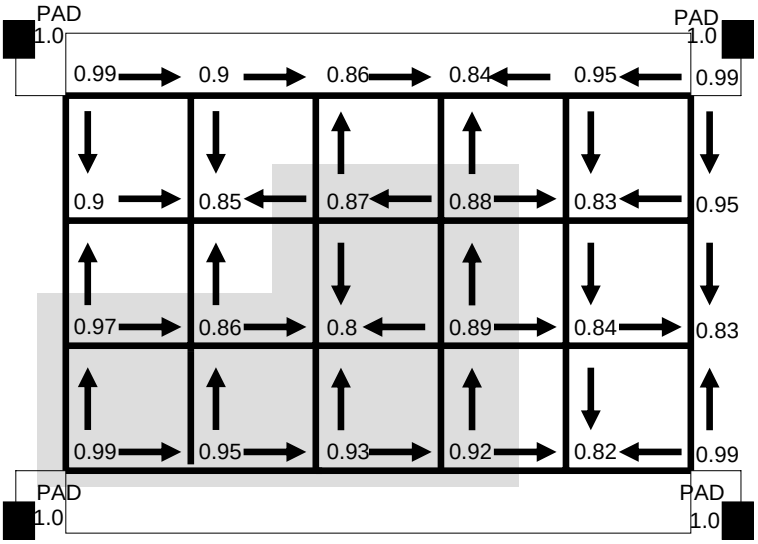

Figure 3.4: ノードの電圧と枝の電流フロー

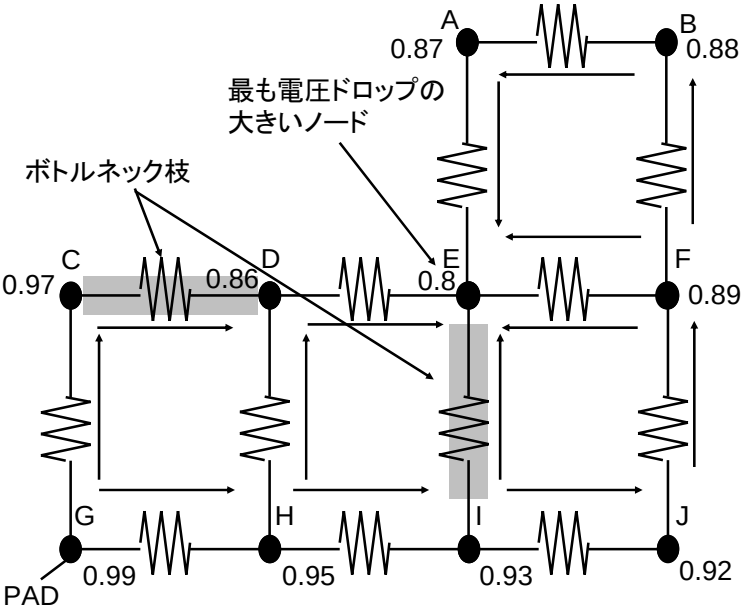

Figure 3.5: ボトルネック枝

これらのパスは、最も電圧ドロップの大きいノードへの電流パスを表す。オーバーフローパス上のすべての枝は、最も電圧ドロップの大きいノードの、電圧を改善できる。オーバーフローパスを抽出することは、提案アルゴリズムのキーで、解の探索空間を減らすことができる。

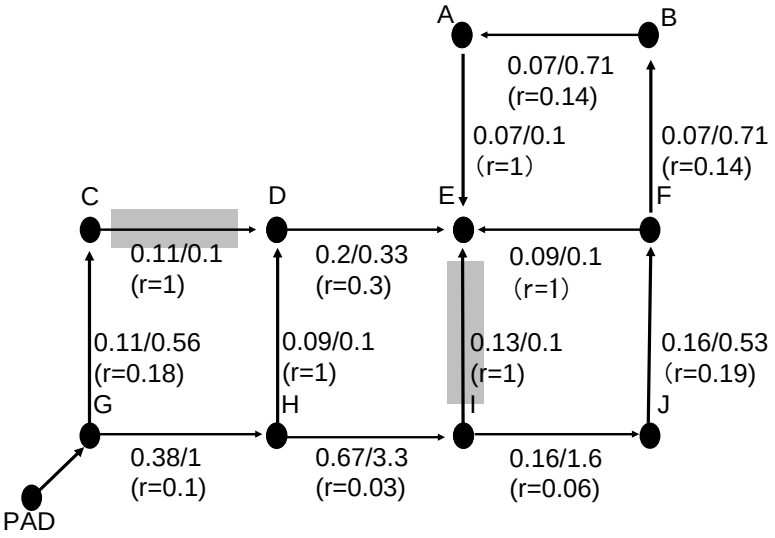

Figure 3.6: オーバーフローパス

最も電圧ドロップの大きいノードから PAD への電流フローを表す電流フロー有向グラフ作成アルゴリズムを述べる（図3.7参照）。最も電圧の低いノードは電源ネットワークグラフを用いて探索され、電流フロー有向グラフ $= (V, E)$ のソースノード S　$\in$ V としてセットされる。そして、深さ優先（幅優先）探索 [27]によって、ソースから PAD ノードへ、電圧が高くなる方向に探索が実行される。同時に有向枝 $\{i, j\} \in$ E が電流フロー有向グラフに追加される。探索は、 PAD ノードに到達すると、処理は終了となる。次に電流フロー有向グラフのシンクノードが作成され、 PAD ノードからシンクノードへの有向枝が作成される。電流フロー有向グラフの有向枝の方向と電流の流れる方向は逆である。図3.7に電流フロー有向グラフの例を示す。

3.5.4　クリティカルパスの抽出

枝の両端の電圧差が大きい枝はノードの電圧ドロップに大きな影響を持つ。最初に、ボトルネック枝を定義する。枝の右のノードの電圧と左のノードの電圧を v_i、 v_j と定義する。以下の条件を満足する枝をボトルネック枝と呼ぶ。

$$|v_i - v_j| \geqq \text{ボトルネック制限値} \tag{3.11}$$

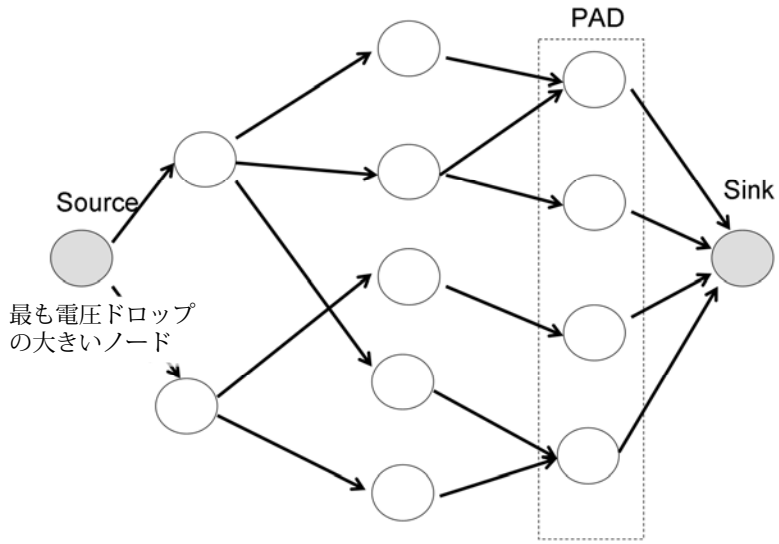

Figure 3.7: 電流フロー有向グラフ

ボトルネック制限値は、電圧ドロップ制約から定義する。ボトルネック枝は電圧を改善するために抵抗を変更すべき枝を示す。加えて、ボトルネック枝は、ネットワークフロー問題に於ける枝の容量がフローを超えている状態と等価である。この時、ボトルネック制限値から決定される容量は、枝の容量とされる。

ボトルネック枝の例を図3.5に示す。ノードの横に記載されている数字は、ノードの電圧を表す。ボトルネック制限値が 0.1 と定義された時、網掛け領域の枝がボトルネック枝となる。ノード D とノード C の間の抵抗が 1Ω の時、ノード D とノード C の間のフローは (0.97-0.86)/1=0.11 となる。また、ノード D とノード C の間の容量は、 0.1/1=0.1 となる。よって、この枝の電流は 0.01 容量を超えている。オーバーフローパス上のボトルネック枝は、電圧改善に効果のある候補枝である。

次に、オーバーフローパスがたくさんある場合、改善すべきパスを 1 本どの様に選択するかについて述べる。クリティカルパスは、最も電圧がドロップしているノードの電圧を改善するのに最も効果のあるパスと定義する。多くのボトルネック枝を持つパスがクリティカルパスと考えることができる。よって、電流フロー有向グラフ上で、小さなコストをボトルネック枝に割り当て、高いコストをほかの枝に割り当てる。トータルコストが小さいパスを見つける。このパスがクリティカルパスである。

クリティカルパス探索方法の詳細について述べる。電流フロー有向グラフの枝は、電圧ドロップの最も大きいノードの電圧を改善する候補である。コストは、枝に容量を超えてフローが流れている枝に小さなコストがセットされ、容量よりフローが小さな枝には高いコストがセットされる。以下にコスト関数の例を示す。

$$cost_{i,j} = B^{(flow_{i,j} - capacity_{i,j})} \quad 0 < B < 1 \tag{3.12}$$

ここで、 $cost_{i,j}$ は、枝 $\{i,j\}$ のコスト、 B は、定数、 $flow_{i,j}$ は枝 $\{i,j\}$ のフローを表し、 $capacity_{i,j}$ は、枝 $\{i,j\}$ の容量を表す。ソースからシンクへのパスで、トータルコストが最小のコストを持つパスが、改善の候補となる。なぜなら、パスのフローが容量を超えているからである。最もコストが小さいパスをダイクストラアルゴリズム [27]によって探す。このパスが、クリティカルパスとなる。図3.6において、パス $\{E,I,H,G\}$ がクリティカルパスとして選択される。

3.5.5 クリティカル枝の抽出

次に、クリティカルパス上の更新すべき枝をどの様に選択するかについて述べる。クリティカルパス上で最もコストが小さい枝が、電圧ドロップを改善する効果が大きい。よって、クリティカルパス上の最もコストが小さい枝を更新する。クリティカル枝は、クリティカルパス上の最もコストが小さい枝として定義する。フロー問題として考えた場合、パス上の最もコストが小さい枝が、フローを妨げる事となる。よって、コストが小さい枝の容量を増加させる。

次に、最も電圧ドロップの大きいノードの電圧を改善するために最も有効なクリティカル枝はネットワークフローとして考えた場合、どの様な意味があるかについて述べる。問題 $A \times V = I$ に於いて、ノードの電圧を解く問題は、それぞれの枝を通過するフローである電流を解く問題として定義できる。枝の右のノードと左のノードの電圧の差が大きい場合、枝を流れる電流に対し、抵抗が大きいと考えることができる。即ち、フローが容量に対して大きいと考えることができる。提案アルゴリズムでは、フローが容量を超えて流れる枝を捜し、枝の容量を増加させる。即ち枝の抵抗を増加させる。結果、電圧ドロップは減少できる。図3.6において、枝の横の数字は枝のフローと容量を表す。図3.6に於いて、網掛け領域の枝は、フローが容量を超えて流れている枝である。枝 $\{E,I\}$ は、図3.6に於いてクリティカル枝として選択される。なぜなら、枝はノードの電圧を改善するのに最も効果のある枝であるからである。

3.5.6 クリティカル枝の抵抗の更新

クリティカル枝の抵抗をどの様に更新するかについて述べる。まず最初に、ターゲットノードの電圧は、電圧ドロップ制約を満足するためにどれくらい改善が行われなければならないかを計算する。この値は V_{update} と定義する。クリティカル枝の両端のノードで電圧差を、V_e と定義する。クリティカル枝を流れる電流を $c_{i,j}$ と定義する。クリティカル枝の抵抗は、以下の式で求めることができる。

$$R_{update} = (V_e - V_{update})/c_{i,j} \tag{3.13}$$

ターゲットノードの電圧ドロップ制約違反を改善する配線幅が計算され、更新される。このアルゴリズムによって、繰り返し回数が削減でき、処理時間が高速化する。

たとえば、図3.6において、ノード E の電圧は、 0.8 から 0.9 に改善すべきである。よって、 V_{update} は、 0.1 である。クリティカル枝 $\{E, I\}$ の V_e は、 0.13 で $c_{E,I}$ は 0.13 である。よって、 R_{update} は (0.13-0.1) / 0.13 である。

3.5.7 電源ネットワーク設計問題の欲張りアルゴリズム

アルゴリズムの詳細を説明する。提案アルゴリズムを以下に示す。

アルゴリズム: 電源ネットワーク設計問題の提案アルゴリズム

Input: $I, V_{min}, V_{max}, G_{max}$
1: すべての枝のアドミッタンスは電源 /GND ネットワーク回路シミュレーションよって計算される。
2: 回路シミュレーションが実行され、各ノードの電圧が計算される。
3: if (ノードの電圧が電圧ドロップ制約を満足しない場合) then {
4: While () {
5: 最も電圧ドロップの大きいノードを探す。
6: 電流フロー有向グラフを作成する。
7: クリティカルパスをダイクストラを用いて電流フロー有向グラフ上で探す。
8: クリティカルパス上の枝から、クリティカル枝を探す。
9: クリティカル枝のアドミッタンスの更新値を求める。
10: If (クリティカル枝のアドミッタンス $\geq G_{max}$) then{
11: クリティカル枝のアドミッタンスを G_{max} とし、この枝をクリティカル枝の候補から外す。
12: }

13: 回路シミュレーションを実行する。ノードの電圧を求める。
14: If (ノードの電圧が電圧ドロップ制約を満足する。もしくはすべての枝のアドミッタンスが $Gmax$) then {
15: 解は、すべての枝のアドミッタンス値。
16: break
17: }
18: }
19: }
Output: すべての枝のアドミッタンス値

---

1 行目に於いて、すべての枝のアドミッタンス値は、電圧ドロップ制約を満足していない解析時間の電源 /GND ネットワーク回路シミュレーションの結果から計算される。 2 行目に於いては、抵抗メッシュモデルの回路シミュレーションが実行され、それぞれのノードの電圧が求められる。 3 行目に於いては、電圧ドロップ制約を満足しているかどうかがチェックされる。もし、満足していれば、処理を終了し、さもなければ、次のステップに進む。 5 行目に於いては、最も電圧ドロップの大きいノードを電源ネットワークグラフ上で探す。 6 行目に於いて、電流フロー有向グラフが、電源ネットワークグラフから作成される。 7 行目に於いて、クリティカルパスが、電流フロー有向グラフ上でダイクストラアルゴリズムを使うことによって求められる。 8 行目に於いて、アドミッタンスが G_{max} より小さいクリティカル枝を捜す。 9 行目に於いて、クリティカル枝の抵抗が更新される。 10 行目に於いて、もしクリティカル枝のアドミッタンスが G_{max} より大きいなら、クリティカル枝のアドミッタンスを G_{max} とする。 13 行目に於いて、回路シミュレーションが実行され、それぞれのノードの電圧が求められる。 14 行目に於いて、電圧ドロップ制約を満足するかどうかがチェックされ、もし満足していたら、処理を終了する。さもなければ、 5 行目へ。

3.6 計算機実験結果

従来アルゴリズム [16, 20] と提案アルゴリズムを電源 /GND ネットワーク設計問題を解く為にプログラムした。従来アルゴリズム [16] は、オンチップデカップリング容量最適化機能を削除し、実行を行った。テーブル 3.1 と図 3.8、 3.9 に従来アルゴリズムと提案アルゴリズムに於いて、内部ノード 9 個と PAD4 個を持ったデータ 1 と電流値のみがデータ 1 と異なるデータ 2 を用いた実験結果を示す。データ 2 のそれぞれの電流値は、データ 1 の電流値を 2 倍にすることによって作成した。それぞれノードの電流値は、 3.3 章で述べた方法によって作成した。

よって、均一ではない。2 章で述べたマクロモデリングアルゴリズムを用い比較を行った。電圧ドロップ制約は 0.1V にセットした。G_{max} は 600,000,000 S、G_{min} は 0.1 S とセットした。すべての枝のアドミッタンスの初期値は、G_{min}、Cu 配線の抵抗値は 1.67E-6Ω-cm、配線厚は、50nm とした。コスト関数 B の値は、0.1、チップサイズは、10mm*10mm とした。図 3.8 は、データ 1 に対し、最大の電圧ドロップを改善するプロセスを示す。[20]と提案アルゴリズムは、電圧ドロップ制約を満足する結果を得ることができた。[20]と提案アルゴリズムに於いて、初期面積を 1 とした場合の面積変化を図 3.9 に示す。これらの結果から、従来アルゴリズム [16]は提案アルゴリズムと比較し、面積増加が急激であることがわかる。加えて、提案アルゴリズムは、データ 2 の解を得ることができたが、従来アルゴリズム [16, 20]は、得る事ができなかった。

テーブル 3.2 に、データ 3、データ 4 を用いた、提案アルゴリズムと従来アルゴリズムでの結果を示す。データ 3 は 100 個の内部ノードと 3 個の PAD を持つデータである。これは実際のデータに近い規模のデータである。実データでマクロモデリングアルゴリズムを適用すると、要求誤差 5% の場合、ノードは約数百ノードに縮退がされる。設計条件はデータ 1,2 と同じである。 初期状態に於いて、82 個の電圧ドロップ違反ノードがあった。従来アルゴリズムと比べ、電源 /GND ネットワークの総配線面積を 1/3 にすることができることを確認した。処理時間は 4m38s で、実用的な処理時間であった。データ 4 は、445 個の内部ノードと 72 個の PAD のデータである。このサイズは、実際のデータに近い規模のデータである。設計条件は、データ 1,2 と同じとした。電源 /GND ネットワーク回路シミュレーション高速化アルゴリズムを適用するために、4 つのドメインに分割し、各ドメインでサブ電源 /GND ネットワーク設計問題とし提案アルゴリズムを適用し、その結果を結合した。従来アルゴリズムと比べ、電源 /GND ネットワークの総配線面積を 1/2 とすることができることを確認した。処理時間は、30m28s で、実用的である。

3.7 まとめ

電圧ドロップ制約を満足し、かつ　電源 /GND ネットワーク配線の総面積を最小化するアルゴリズムを提案した。提案アルゴリズムはネットワークフローアルゴリズムを用いている。電圧ドロップに最も影響のある枝を探し、その枝の抵抗を変更することによって、電圧ドロップを改善する。提案アルゴリズムは、電圧ドロップに最も影響のある枝を更新するため、有効な方法である。

Table 3.1: 小データを用いた従来アルゴリズムと提案アルゴリズムの比較

	[20]		[16]		Proposed	
	#iter	worst voltage drop	#iter	worst voltage drop	#iter	worst voltage drop
data1	44	0.10V	1000	0.25V	85	0.10V
data2	100	0.16V	1000	0.52V	105	0.10V

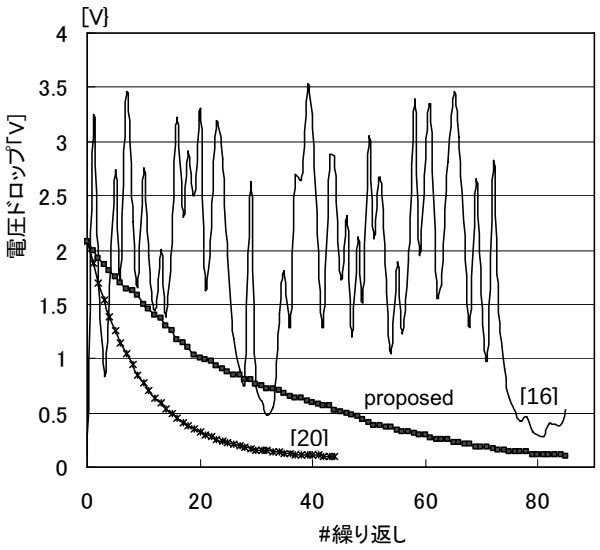

Figure 3.8: 小データ 1 を用いた電圧改善の過程

計算機実験結果から、従来アルゴリズムと比べ、電源 /GND ネットワーク配線の総面積を 1/3 にすることができた。また、電圧ドロップ制約を満足する解を、従来アルゴリズムでは見つけることができなかったデータに対しても求めることができた。

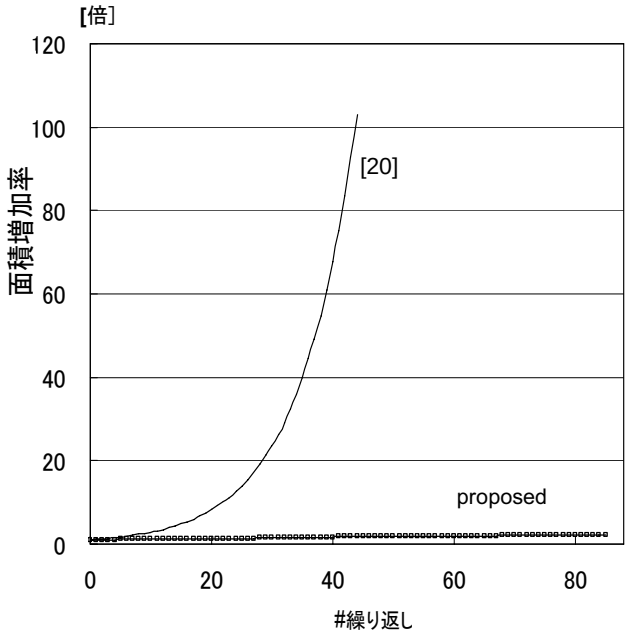

Figure 3.9: 小データ 1 を用いた配線の総面積の増加の過程

Table 3.2: 実データ規模のデータ 3,4 を用いた従来アルゴリズムと提案アルゴリズムの比較結果

	[20]		Proposed	
	CPU	power/ground wiring area ratio	CPU	power/ground wiring area ratio
data3	53.4s	1.5242	4m38s	1.0004
data4	22m57s	3.1661	30m28s	1.2567

Chapter 4

オンチップデカップリング容量最適化アルゴリズム

4.1 概要

プロセス技術の進歩に伴い、トランジスタの動作周波数は上がり、 1 チップに搭載される回路の数も増加した。その結果、電力消費量とノイズは大きく増加した。加えて、電源電圧は、消費電力を抑えるために、低下した。即ち、信号線の振幅は、小さくなった。そのため、電圧ドロップの信号線への影響が大きくなった。よって、電圧ドロップの問題はより、重要となった。加えて、オンチップデカップリング容量の追加は、電圧ドロップを解決するために重要な手法である。オンチップデカップリング容量はチップ面積に大きな影響がある。よって、製造コストの面から、オンチップデカップリング容量追加の総量は最小限に押さえなくてはならない。加えて、テクノロジーの進歩に伴い、リーク電流は指数関数的に増加し、消費電力の主原因となってきている。リークによる消費電力を減少させるためには、オンチップデカップリング容量の総量を減らすことが重要である。オンチップデカップリング容量の総量を減少させることは、動作時の消費電力を減少させるのみならず、スタンバイ電力を減少させる。この点から、オンチップデカップリング容量の総量を減少させることは重要である。そこで、電圧ドロップ制約を満足し、オンチップデカップリング容量の総量を最小化するアルゴリズムを提案する。

最も用いられているアルゴリズムは、電圧ドロップが大きい問題の箇所に近いドメインにオンチップデカップリング容量を追加する方法 [30], [31], [32], [28], [16]である。オンチップデカップリング容量はノイズソースの近くに配置すると、ノイズ抑制に有効に働く。もし、オンチップデカップリング容量と大きな電流を消費するドメインの距離が離れていたら、チャージの移動が十分早く行われない。このため、オンチップデカップリング容量追加の効果は小さくなる。加えて、オンチップデカップリング容量を配置することが出来る空いている領域は、電圧ドロップの大きい問題のドメイン内に十分にない可能性もある。

近年、感度ベースのアルゴリズム [29], [30], [31], [32] が提案されている。このアルゴリズムでは、オンチップデカップリング容量を追加できるすべての候補に対し、感度を計算する必要がある。感度の計算には、すべての候補位置の波形を、オンチップデカップリング容量追加前と追加後で保持する必要がある。そのため、非常に多くのメモリを必要とする。加えて、1 つのオンチップデカップリング容量の追加に対し、1 回のトランジェント解析を行う必要があり、非常に多くの解析時間が必要である。他の方法として、チャージベースのアルゴリズム [28], [16], [33]が提案されている。チャージベースアルゴリズムは最適化の間、トランジェント解析を行う必要がない。よって、処理時間の点で感度ベースのアルゴリズムと比べ優れている。アルゴリズムの詳細を以下に示す。

STEP1 寄生容量成分を含んだ電源 /GND ネットワークでトランジェント解析を行う。

STEP2 ノードの電圧が、電圧ドロップ制約以内かどうかチェックする。そして、そのドメインに電圧ドロップ制約違反があるのかを調べる。もし、電圧ドロップ制約違反がなければ最適化を完了する。さもなければ STEP3 へ。

STEP3 各電圧ドロップ制約違反ドメインに対し最適なオンチップデカップリング容量の追加を行うために線形計画法と電源 /GND ネットワーク回路シミュレーション高速化アルゴリズム [19]を適用する。

STEP4 電圧ドロップ制約違反ドメインにオンチップデカップリング容量を配置し、 STEP1 へ。

これらのアルゴリズムは、逐次線形計画法（Successive Linear Programming (SLC)）を用いている。多くの逐次線形計画を解くプログラム [17], [18] が公開されている。しかし、このアルゴリズムは SoC 設計のような大規模回路に適用することは難しい。そこで、電圧ドロップ制約違反のドメイン毎に線形計画法を適用し、オンチップデカップリング容量を追加する。このアルゴリズムは、緩和を用い解の探索範囲を制限するため、精度は悪化する。

本論文では、ネットワークフローアルゴリズム [21]を用いたアルゴリズムを提案する。提案のアルゴリズムは、オンチップデカップリング容量をクリティカルノード、すなわち、電圧ドロップに最も影響のあるノードにのみ追加する。改善効果を計算し、オンチップデカップリング容量追加の効果が最も大きく、ノイズ源の近くのノードに追加されるので、オンチップデカップリング容量の増加を最小化することができる。よって、提案アルゴリズムは、電圧ド

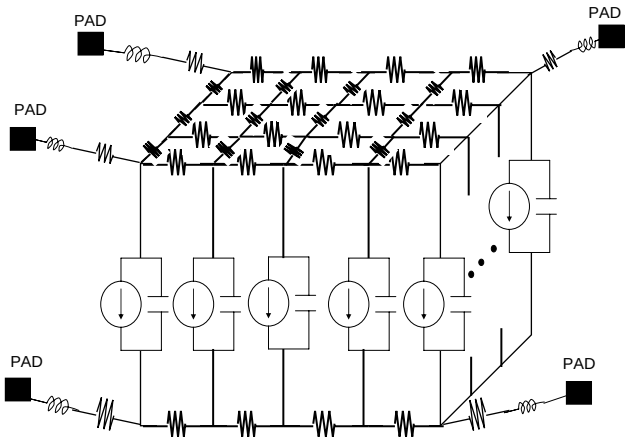

Figure 4.1: 電源 /GND ネットワークモデル

ロップ制約を満足し、オンチップデカップリング容量の総量を最小化し、高速に解を得ることができる。本アルゴリズムでは、オンチップデカップリング容量は、電圧ドロップ制約違反の箇所の近くではなく、実際にオンチップデカップリング容量が必要なノイズソースの近くに置かれる。

4.2 準備

4.2.1 電源 /GND ネットワークモデル

図4.1に、電源 /GND ネットワークモデルを示す。ここで、電源グリッドの各抵抗は各層の配線の合成抵抗で容量は、配線の寄生容量とオンチップデカップリング容量との和である。説明を簡単にする目的で、電源 /GND のグリッドは等間隔で作成する。通常、電源 /GND ネットワーク回路シミュレーションでは、チップのインダクタンスは考えない。なぜなら、チップのインダクタンスの値は小さく、回路シミュレーション結果に対する影響は小さいからである。そこで、本論文でもチップ内におけるインダクタンスは考慮しない。パッケージのインダクタンスは考慮する（図4.1）。

4.2.2 違反ウインドウ

図 4.2はノードの電圧波形の例を表す。図4.2に於いて, 電圧ドロップ制約は、 $1.0V \pm 0.1$ 以内である。違反ウインドウは、電圧波形から算出する。まず、電圧ドロップが最大の時間を見つける。次に最初に電圧ドロップ制約を越えた時刻を特定する。この時間を Ts とする。

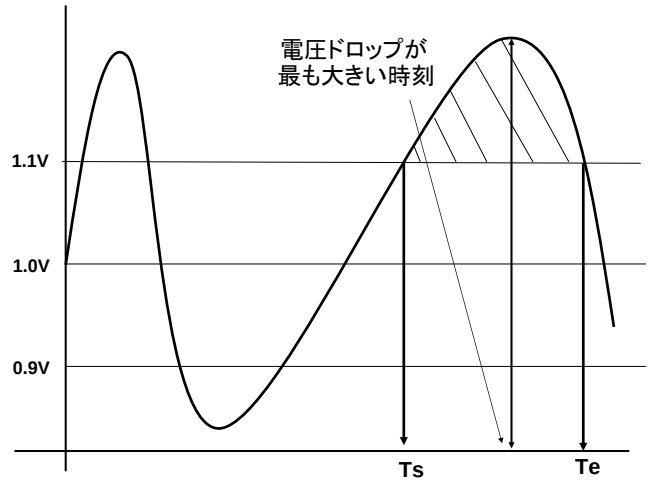

Figure 4.2: あるノードの違反ウインドウ

次に、電圧が、電圧ドロップ制限を最初に下回った時刻を特定する。この時間を Te とする。違反ウインドウは、 [Ts Te] と表現できる。オンチップデカップリング容量を追加する目的は、電圧を改善するためである。電圧ドロップ制約を満足するために、電圧ドロップ制約を満足しない領域（図中の網掛け）に相当する電荷をオンチップデカップリング容量を用いて供給できれば良い。

4.2.3 電源 /GND ネットワークの抵抗メッシュモデル

電源 /GND ネットワークの抵抗メッシュモデルは、電圧ドロップを満足していない解析時間の回路シミュレーションの結果から作成される。枝の抵抗は、枝に流れる電流とノードの電圧から計算される。ノードの電流は、電源ノードから GND ノードへ流れる電流とする。パッケージのインダクタンスは抵抗として表現され、チップの容量は電流として表現される。図 4.3は、電源 /GND ネットワークの抵抗メッシュモデルを示す。以下に、オンチップデカップリング容量最適化問題を定式化する時に使う記号を定義する。(N_u, E_b) は、電源 /GND ネットワークの抵抗メッシュモデルを表す。以下に本論文で用いる記号を定義する。

N_d: 電源 /GND ネットワークのノードの集合
N_s: PAD のノードの集合
$N_u = N_d \cup N_s$

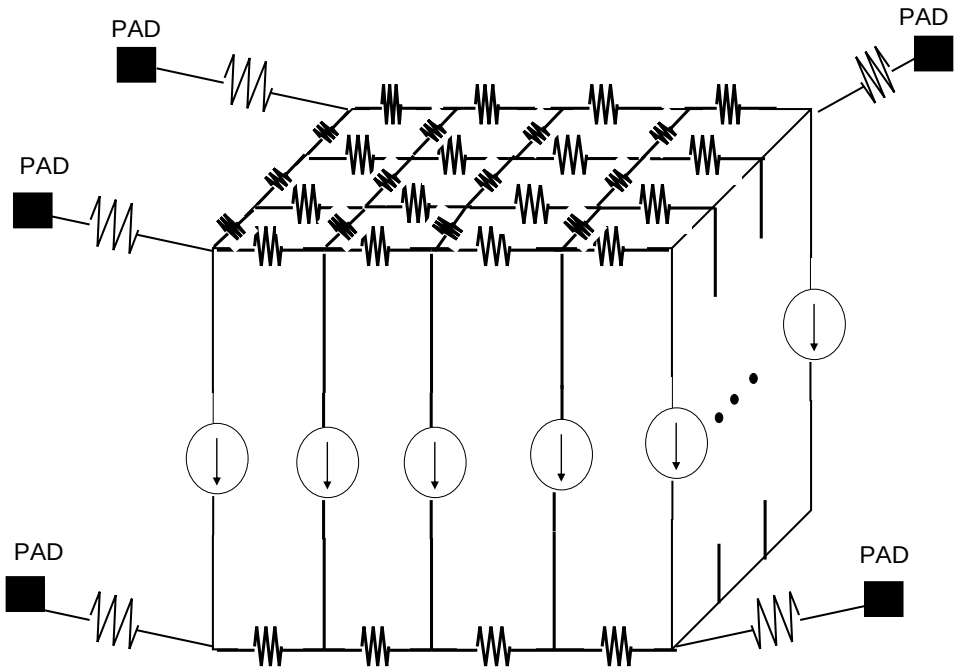

Figure 4.3: 電源 /GND ネットワークの抵抗メッシュモデル

N_p: 電源ネットワークのノードの集合

N_g: GND ネットワークのノードの集合

$N_d = N_p \cup N_g$

$n = |N_u|$

E_p: N_d のノードと N_s のノードを結ぶ枝の集合

E_g: N_d のノード間を結ぶ枝の集合

$E_b = E_p \cup E_g$

$c_{i,j}$: 枝 $\{i, j\}$, $i, j \in N_u$ の電流

v_i: ノード $i, i \in N_u$ の電圧

$g_{i,j}$: 枝 $\{i, j\}$, $i, j \in N_u$ のアドミッタンス

$r_{i,j}$: 枝 $\{i, j\}$, $i, j \in N_u$ の抵抗

i_i: ノード $i, i \in N_u$ の電流源

4.3 オンチップデカップリング容量最適化問題の定式化

4.3.1 電圧ドロップ制約

デジタル論理が正しく動作するためには、電圧ドロップ制約を満足しなくてはならない。この電圧ドロップ制約は、STA(Static Timing Analysis) システムで加味する電圧ドロップ値から決定される。V_{minp} と V_{maxp} は、電源ネットワークの最小許容電圧と最大許容電圧と定義する。V_{ming} と V_{maxg} は、GND ネットワークの最小許容電圧と最大許容電圧と定義する。よって、電圧ドロップ制約は以下の様に定義することができる。

$$V_{minp} \leqq v_i \in N_p \leqq V_{maxp} \tag{4.1}$$

$$V_{ming} \leqq v_i \in N_g \leqq V_{maxg} \tag{4.2}$$

4.3.2 回路制約

ノードの電圧の集合は、ノードの電圧のベクトル $V = \{v_1, v_2, , v_n\}^T$ として定義できる。ノードの電流の集合は、ノードの電流のベクトル $I = \{i_1, i_2, , i_n\}^T$ として定義できる。キルヒホッフの電流則を用いると、回路方程式を作成することができる [25]。この回路制約は以下で表すことができる。

$$A \cdot V = I, \tag{4.3}$$

ここで、 A は $n \times n$ アドミッタンスマトリックス。詳細な回路制約作成方法は、[25, 26]で述べられている。

4.3.3 オンチップデカップリング容量制約

各ドメインでオンチップデカップリング容量を配置できる領域には限りがある。よって、オンチップデカップリング容量の面積には最大値の制約がある。$Q_{max,i}$ を、ドメイン i に配置可能オンチップデカップリング容量として定義する。よって、オンチップデカップリング容量の面積制約は以下で表すことができる。

$$q_i \leqq Q_{max,i} \tag{4.4}$$

ここで、 q_i は、ドメイン i のオンチップデカップリング容量の総量を表す。

4.3.4 目的関数

オンチップデカップリング容量の総量を最小化すると、リーク電流や、スタンバイ電流を低減することができる。よって、目的関数は、電源 /GND ネットワークのオンチップデカップリング容量の総量を最小化することである。オンチップデカップリングの総量は、以下で表すことができる。

$$\sum_{i \in D} q_i \tag{4.5}$$

ここで D は、ドメインの集合を現す。よって、目的関数は、以下で定義できる。

$$Obj : \min \sum_{i \in D} q_i \tag{4.6}$$

4.3.5 オンチップデカップリング容量最適化問題

以上の結果から、オンチップデカップリング容量最適化問題は、以下で定式化できる。

$$\begin{aligned} Minimize \quad & \textstyle\sum_{i \in D} q_i \\ Subject \ \ to \quad & V_{minp} \leqq v_i \in N_p \leqq V_{maxp} \\ & V_{ming} \leqq v_i \in N_g \leqq V_{maxg} \\ & q_i \leqq Q_{max,i} \\ & A \cdot V = I \end{aligned} \tag{4.7}$$

本論文では、オンチップデカップリング容量最適化問題を解くためのアルゴリズムを提案する。この問題を解くことによって、電圧ドロップ制約を満足し、かつ、オンチップデカップリング容量の総量を最小化することが出来る。

4.4 オンチップデカップリング容量最適化アルゴリズム

4.4.1 提案アルゴリズムの概要

提案アルゴリズムのアウトラインを示す。提案アルゴリズムは、まず寄生の容量のみから処理を始める。オンチップデカップリング容量の総量を最小化するために、電圧改善に効果がある箇所に対してのみオンチップデカップリング容量を追加することによって、電圧ドロップ制約を満足する解を得る。

電源 /GND ネットワークモデルを、マクロモデリングアルゴリズムを用いて作成する。
電源 /GND ネットワーク回路シミュレーションを実行する。
While（電圧ドロップ制約を満足していない解析時間がある。 ）
{
電圧ドロップを満足していない解析時間に対して、電圧ドロップに
最も影響のあるパスを見つけ、そのパス上のノードに電流源（オンチップ
デカップリングを追加する代わりに）を追加する。
この処理を電圧ドロップ制約を満足するまで繰り返す。
各ドメインに電流源に対応するオンチップデカップリング容量を追加する。
電源・ GND ネットワーク回路シミュレーションを実行する。
}

以降では、上記アルゴリズム中の電圧ドロップに最も影響のあるノードを見つけてオンチップデカップリング容量を追加する部分の電圧ドロップ改善アルゴリズムをより詳細に説明する。

4.4.2 電圧ドロップ改善アルゴリズム

ネットワークフローアルゴリズムを用いた電圧ドロップ制約改善アルゴリズムを述べる。提案アルゴリズムは、以下の手順を繰り返し行う。

STEP1 電源 /GND ネットワークの抵抗メッシュモデルを、電圧ドロップ制約を満足していない解析時間の電源 /GND ネットワーク回路シミュレーションの結果から作成する。

STEP2 抵抗メッシュモデルの回路シミュレーションが実行され、ノードの電圧が計算される。

STEP3 最も電圧ドロップの大きいノードを見つける。

STEP4 最も電圧ドロップの大きいノードの電圧を決定するパス（オーバーフローパスと呼ぶ）を見つける。

STEP5 オーバーフローパスに含まれるパスで、最も電圧のドロップしているノードの電圧に最も影響あるパス（クリティカルパスと呼ぶ）を見つける。

STEP6 クリティカルパス上のノードで最も電圧がドロップしているノードの電圧決定に最も影響のあるノード（クリティカルノードと呼ぶ）を見つける。

STEP7 クリティカルノードの電流源に電流を追加し、STEP2 へ

提案アルゴリズムでは、電圧ドロップ制約違反ノードの改善に最も効果のあるパスを見つけ、電圧ドロップの最も大きいノードの電圧に最も影響のあるパス上のノードの電流源にのみ電流を追加する。よって、電流源に追加する電流の量（オンチップデカップリング容量）を最小に押さえることができる。加えて、DC 解析を用いるので、処理時間を抑えることが出来る。

我々はオーバーフローパスとクリティカルパスを特定するアルゴリズムを開発した。まず、アルゴリズムの詳細を説明する。次に、クリティカルノードの算出アルゴリズムを説明する。最後に、オンチップデカップリング容量の追加アルゴリズムを説明する。

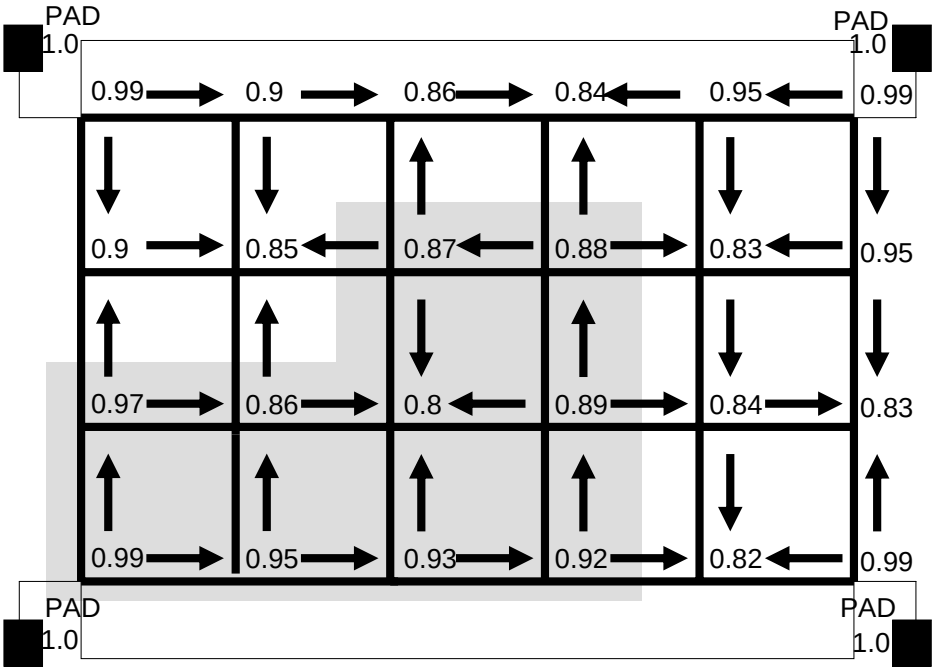

Figure 4.4: ノードの電圧と枝の電流フロー

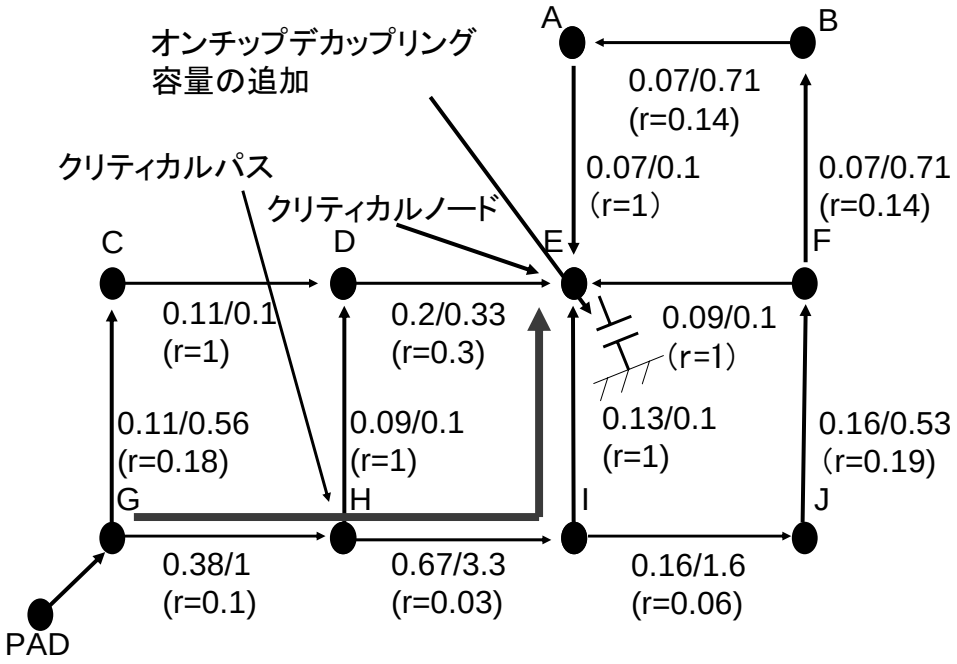

Figure 4.5: クリティカルパスとクリティカルノード

4.4.3 オーバーフローパスの抽出

この章では、最も電圧ドロップの大きいノードの電圧を決定するオーバーフローパスを見つける方法について述べる。ノードの電圧は、電流と PAD からノードへのパスの抵抗で決まる。電流は電圧の高いノードから電圧の低いノードへ流れる。電流の経路はパスである。よって、最も電圧の低いノードからの電流フローを探索し、パスを見つける。オーバーフローパスは、 PAD から最も電圧の低いノードへの電流パスとして定義することができる。

オーバーフローパス算出の過程を図 4.4、4.5を用いて説明する。図4.4で、ノードの横の数字は、ノードの電圧を表す。矢印は電流の流れる方向を示す。図4.4において、 0.8 は最も電圧が小さい。そこで、電圧が 0.8 のノードが選ばれる。網掛け領域のパス（図4.4）は電流フローパスとして選択される。これらのパスは最も電圧の低いノードの電圧を決定するので、このパスがオーバーフローパスである。図4.5は算出を行った結果であり、 $\{E, A, B, F, J, I, H, G\}$, $\{E, F, J, I, H, G\}$, $\{E, I, H, G\}$, $\{E, D, H, G\}$, $\{E, D, C, G\}$ がオーバーフローパスである。

電圧の最も低いノードから PAD ノードまでの電流フローを表す電流フロー有向グラフの作成アルゴリズムについて説明を行う。最も電圧の低いノードが、電源 /GND ネットワー

クモデルを用いて見つけられ、電流フロー有向グラフ = (V, E) のソースノードとして s $\in$ V セットされる。電源 /GND ネットワークモデルに於いて、最も電圧の低いノードはソースノードとして設定され、深さ優先探索 [27] によって、ソースノードから PAD ノードへ、電圧が大きくなる方向へ、探索が実行される。同時に、電流フロー有向グラフの有向枝 $\{i, j\} \in$ E が作成される。 PAD に到達した時、処理は終了する。次に、電流有向グラフのシンクノードが作成され、 PAD ノードからシンクノードへの有向枝が作られる。

4.4.4 クリティカルパスの抽出

両端の電圧が大きく異なる枝は、電圧ドロップ制約違反ノードに大きな影響がある。最初に、ボトルネック枝を定義する。枝の右ノードの電圧と左のノードの電圧を v_i 、 v_j と定義する。以下の条件を満足する枝をボトルネック枝と呼ぶ。

$$|v_i - v_j| \geqq \text{ボトルネック制限値} \tag{4.8}$$

ボトルネック制限値は、電圧ドロップ制約から定義される。ボトルネック制限値から決定された容量を、枝の容量として定義する。ボトルネック枝は、フロー問題において、電流フローが枝の容量を超えていることを表す。

ボトルネック枝の例を図4.5に示す。ボトルネック制限値が 0.1 にセットされた時、枝 {C, D} と 枝 {E, I} は、ボトルネック枝である。ノード D とノード C の間の抵抗が 1Ω の時、ノード D とノード C の間のフローは、 (0.97-0.86)/1=0.11 で、ノード D とノード C 間の容量は 0.1/1=0.1 である。よって、枝の電流は、容量を 0.01 を超えて流れる。

次に、改善すべきパスを選択する方法について述べる。クリティカルパスは、最も電圧ドロップの大きいノードの改善に最も影響があるパスと定義する。沢山のボトルネック枝を持つパスをクリティカルパスと考えることができる。よって、電流フロー有向グラフにおいて、小さいコストをボトルネック枝に割り当て、他の枝に大きなコストが割り当てる。電流フロー有向グラフ上で最もコストの合計が小さいパスを見つける。これがクリティカルパスである。

電流フロー有向グラフを用いて、クリティカルパスをどのように見つけるかについて詳細に述べる。容量を超えて電流が流れる枝に小さなコストを、容量以内で電流が流れている枝に大きなコストを枝にセットする。以下にコストの関数の例を示す。

$$cost_{i,j} = B^{(flow_{i,j} - capacity_{i,j})} \quad 0 < B < 1 \tag{4.9}$$

ここで $cost_{i,j}$ は枝 $\{i,j\}$ のコスト、 B は定数、 $flow_{i,j}$ は枝 $\{i,j\}$ のフロー、 $capacity_{i,j}$ は、枝 $\{i,j\}$ の容量を表す。ソースからシンクへのパスに於いて、トータルコストが最も小さいパスが、改善候補となる。なぜなら、パスのフローが容量を超えているパスであるからである。よって、ダイクストラアルゴリズム [27] を用いて最もコストの合計が小さいパスを見つける。図4.5で、枝の横の数字は、枝のフローと容量を表す。図4.5で、パス $\{E,I,H,G\}$ は、クリティカルパスとして選択される。

4.4.5 クリティカルノードの抽出

電圧改善に最も効果のあるクリティカルパス上のノードを算出し、オンチップデカップリング容量をそのノードに追加する方法について述べる。クリティカルノードは、クリティカルパス上のノードで電圧改善に最も効果のあるノードと定義する。図 4.6、 4.7 を用いて、オンチップデカップリング容量を追加する効果について説明を行う。図 4.6 はクリティカルパスを表す。最も電圧ドロップの大きいノードの電圧は、以下で決定される。

$$\sum_{\{i,j\}\in CP} r_{i,j} \times c_{i,j} \tag{4.10}$$

ここで、 $r_{i,j}$ は、枝 $\{i,j\}$ の抵抗、 $c_{i,j}$ は、枝 $\{i,j\}$ の電流、 CP は、クリティカルパス上の枝の集合を表す。

図 4.7 に於いて、もし、オンチップデカップリング容量がノード 4 に追加された場合、電流は、ノード 4 に追加されたオンチップデカップリング容量から、ノード 5， 6 に供給される。よって、最もドロップの大きいノードの電圧は、 $r_{1,2} * capc_4 + r_{2,3} * capc_4 + r_{3,4} * capc_4$ 改善される。ここで、 $capc_i$ は、ノード i に追加されたオンチップ容量から供給される電流である。提案アルゴリズムでは、コストは、オンチップデカップリング容量を追加することによって改善される電圧によって定義する。加えて、オンチップデカップリング容量は、ノイズソースの近くに置かれることによって、ノイズを抑制する効果がある (図4.8)。よって、オンチップデカップリング容量は、電流ソースの近くに追加すべきである。ノード i のコストの例を以下に示す。

$$cost_{update} = \sum_{\{i,j\}\in UP} r_{i,j} \times capc_i \tag{4.11}$$

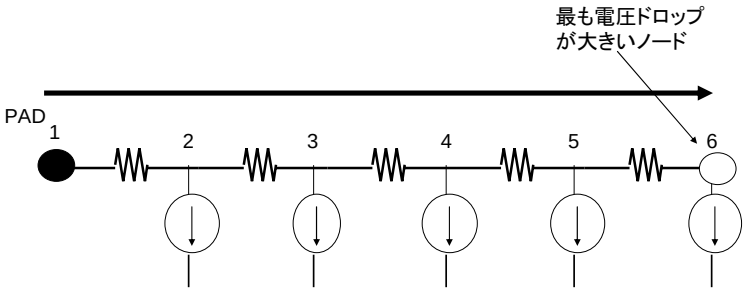

Figure 4.6: オンチップデカップリング容量追加前

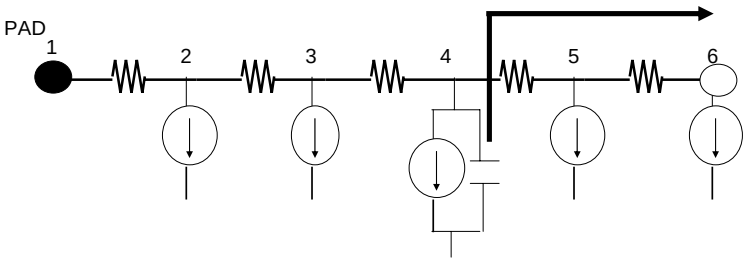

Figure 4.7: オンチップデカップリング容量追加後

ここで、 UP は、ノード i より PAD 側の枝の集合で、クリティカルパス上の枝の集合を表す。 $capc_i$ は以下で計算される。

$$capc_i = \alpha \times c_{i,j} + \beta \times i_i \tag{4.12}$$

ここで、 α と β （$\alpha+\beta=1$）は定数、 $c_{i,j}$ は、ノード i より PAD 側にある枝 $\{i,j\}$ の電流を表す。オンチップデカップリング容量は、最もコストの大きいノードに追加される。

4.4.6 オンチップデカップリング容量の追加

クリティカルノードに追加を行うオンチップデカップリング容量の量を如何に決定するかについて述べる。最初に、電圧ドロップ制約を満足するために最もドロップが大きいノードの電圧をどれ位改善しなくてはならないかを求める。この量は、 V_{update} と定義する。クリティ

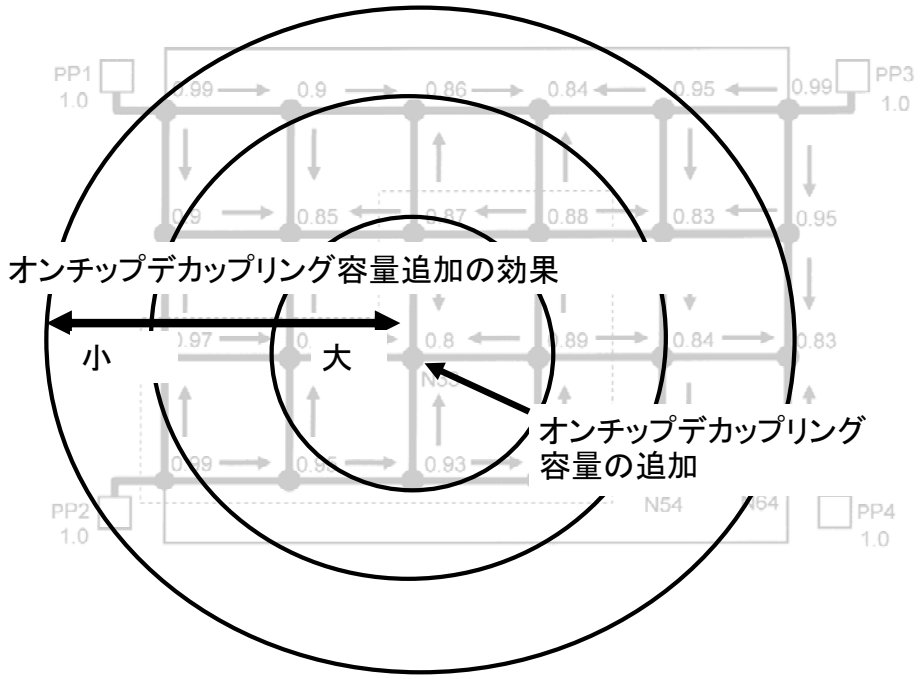

Figure 4.8: オンチップデカップリング容量の有効エリア

カルノードから PAD へのパスの抵抗は、 R_{update} と定義され、電圧改善のための電流値は、C_{update} と定義される。ノード i に追加される C_{update} は以下で計算することができる。

$$C_{update} = \min\{V_{update}/R_{update}, c_{i,j}\} \tag{4.13}$$

電流 C_{update} は、電源 /GND ネットワークの電流メッシュモデルのクリティカルノードに追加される。これに続き、電源 /GND ネットワークの抵抗メッシュモデルの回路シミュレーションが実行される。この処理は、電圧ドロップ制約違反ノードが無くなるまで繰り返される。

もし、電圧ドロップ制約違反ノードが無くなったら、オンチップデカップリング容量は、電流値を元に、以下の式を用いて追加されて、電源 /GND ネットワークのそれぞれのノードに追加される。

$$cap_{update} = \gamma \int_{t_s}^{t_e} ii_i \cdot dt \tag{4.14}$$

ここで、ii_i は、ノード i の電流源に追加された電流、γ は定数、$[t_s, t_e]$ は、違反ウインドウもしくはタイムステップを表す。これに続いて、電源 /GND ネットワーク回路シミュレーションが実行され、ノードの電圧が電圧ドロップ制約以内かどうかがチェックされる。もし、電圧ドロップ制約違反ノードが無ければ、最適化は、完了する。電圧ドロップ制約違反があった場合、電圧ドロップ改善アルゴリズムが再度、電圧ドロップ制約違反ノードが無くなるまで実行される。

図 4.5では、ノード E はクリティカルノードとして選択される。なぜなら、ノードは最も電圧ドロップの大きいノードの電圧を改善するのに最も有効であるからである。よって、オンチップデカップリング容量は、ノード E に追加される。

4.5 計算機実験結果

従来のアルゴリズム [28], [16], [33] と提案アルゴリズムをオンチップデカップリング容量最適化問題のためにプログラムした。すべての実験は 2.6GHz リナックスマシン、4 GB メモリ搭載マシンで行った。テーブル 1 と図 4.9、4.10 は、従来アルゴリズムと提案アルゴリズムを実際のデータ（約 3 億インスタンス）を用いて比較を行った結果を表す。データの消費電力は、約 1 ワットであった。データ 2 の寄生容量は 0F、データ 1,3 の寄生容量は 2nF とした。電圧ドロップ制約は 0.1V とした。すべてのノードのオンチップデカップリング容量の初期値は、0 とした。トランジェント解析を 10ns 実行することとした。電源 /GND ネットワークモデルのそれぞれのグリッドの抵抗は、同じ値にセットした。

最初に、2 章で述べたマクロモデリングアルゴリズムを用いて、電源 /GND ネットワークモデルを作成した。よって、最適化のための電源 /GND ネットワークモデルのノードの数は、約 100 個であった。両アルゴリズムは最適化のアルゴリズムを公平に比較するために、電源 /GND ネットワークモデルに対し適用した。マクロモデリングアルゴリズムを用いているので、電圧ドロップ解析の誤差を見積もることが出来る。電圧ドロップ制約は誤差を加味して設定した。この条件で最適化を実行した。よって、デザイン制約を満足しつつ、最適化アルゴリズムの速度を上げることができた。

テーブル 1 において、decap は、アルゴリズムによって追加されたオンチップデカップリング容量の総量を表す。また、CPU は、電源 /GND ネットワークモデル作成の時間を含まない CPU 時間を表す。テーブル 1 から、提案アルゴリズムは、従来アルゴリズムを用いた場合と比較し、追加されるオンチップデカップリング容量の量が少なくすむことが分かる。図

Table 4.1: 従来アルゴリズムと提案アルゴリズムの比較結果

	[28], [16], [33]		Proposed		
	CPU	decap	CPU	decap	△ decap
data1	1m44s	3.08nF	3m38s	1.9nF	−38.3%
data2	2m55s	452pF	7m32s	231.6pF	−49%
data3	1m39s	3.34nF	3m28s	1.59nF	−52.4%

(HP Linux workstation 2.6GHz CPU 4GB memory)

4.9 は、データ 1 での最大ドロップの改善の過程を示した図である。 x 軸は、繰り返し数、 y 軸は電圧ドロップと電圧ドロップ制約の差を示す。従来アルゴリズム、提案アルゴリズム共に、電圧ドロップ制約を満足する結果を得ることができた。従来アルゴリズムと提案アルゴリズムで、最適化はトランジェント解析結果を元に実行された。最適化が実行された後、再度トランジェント解析を実行し、この結果をグラフにプロットした。図 4.10 は、データ 1 でのオンチップデカップリング容量の追加の過程を表す。従来アルゴリズムと比べ、オンチップデカップリング容量の総量を 38.3% に減少させることができた。また、 CPU 時間は 3m38s と実用的であった。データ 2 において、従来アルゴリズムと提案アルゴリズムは、電圧ドロップ制約を満足する結果を得ることができた。従来アルゴリズムと比べ、オンチップデカップリング容量の総量を 49% 削減することが出来た。 CPU 時間は 7m32s と実用的であった。データ 3 で、従来アルゴリズム、提案アルゴリズムで、電圧ドロップ制約を満足する結果を得ることができた。従来アルゴリズムと比べ、オンチップデカップリング容量の総量を 52.4% 削減することが出来た。 CPU 時間は 3m28s と、実用的であった。

4.6 まとめ

電圧ドロップ制約を満足し、オンチップデカップリング容量の総量を最小化する電源 /GND ネットワークのためのアルゴリズムを提案した。提案アルゴリズムはネットワークアルゴリズムを用いている。電圧ドロップに最も影響のあるパスを見つける。電圧ドロップを、そのパス上の最も電圧ドロップ改善に効果のあるノードにオンチップデカップリング容量を追加することによって改善する。提案アルゴリズムは、電圧ドロップに最も影響のあるノードにオンチップデカップリング容量を追加するので、有効なアルゴリズムです。

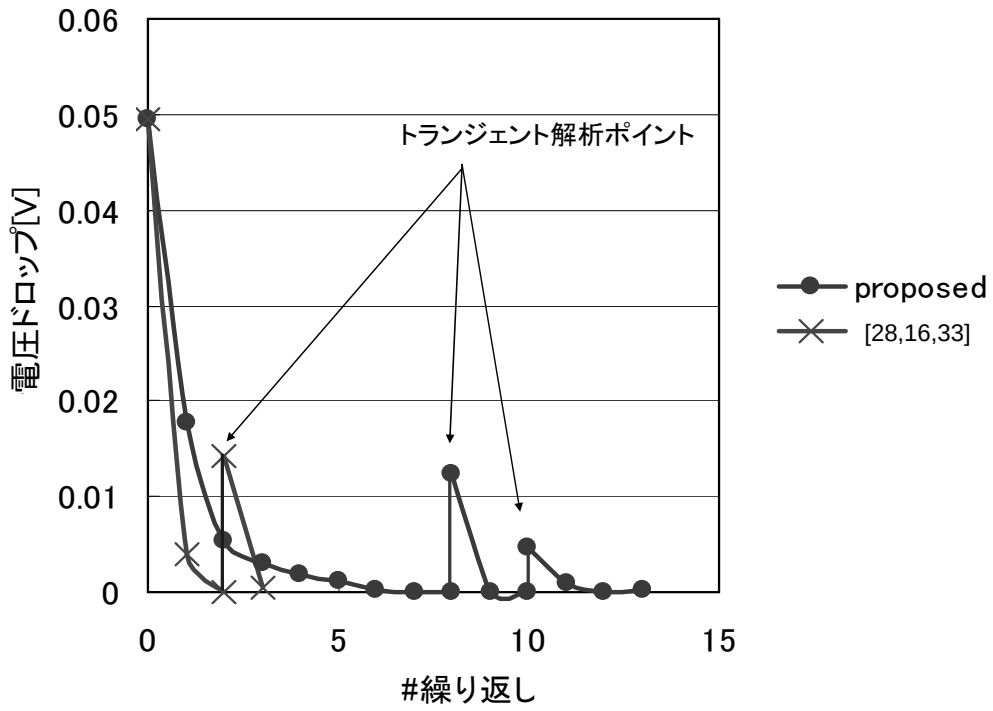

Figure 4.9: 電圧ドロップ改善の過程

計算機実験結果から、従来アルゴリズムと比べ、電源 /GND ネットワークのオンチップデカップリング容量の総量を約 1/2 にすることができた。また、電圧ドロップ制約も満足することができた。

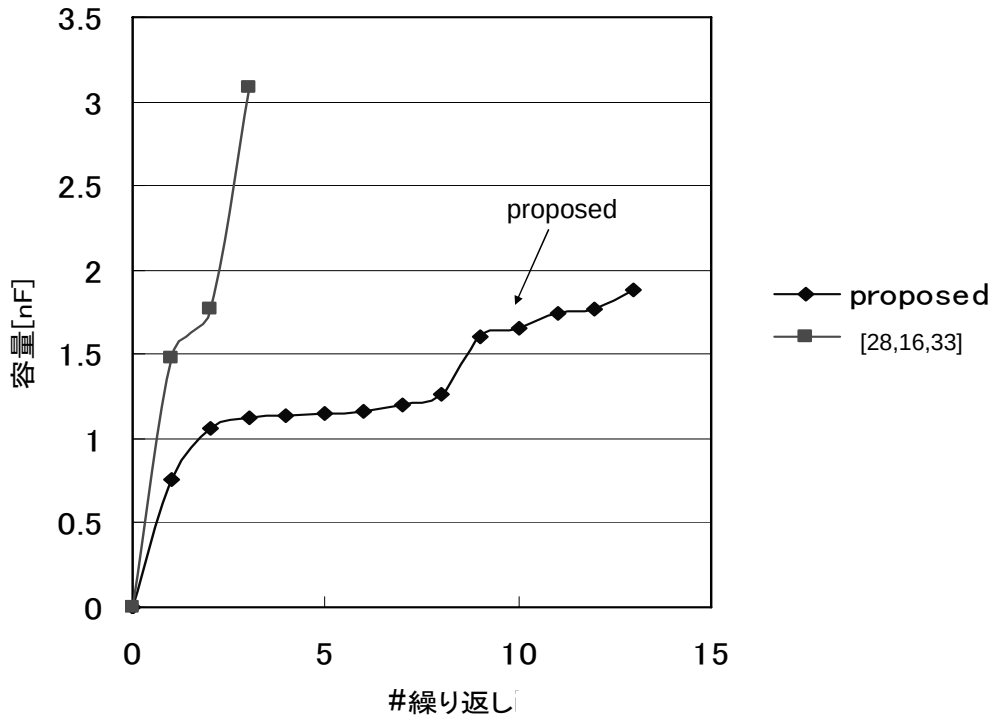

Figure 4.10: オンチップデカップリング容量追加の過程

Chapter 5

基板ノイズ最適化アルゴリズム

5.1 概要

プロセス技術の進歩に伴い、同じ LSI 上に多くのデジタル回路、アナログマクロを搭載するようになった。特に、近年、 USB、 LVDS、 HDMI といった高速 IO インターフェースマクロのみでなく、高性能 ADC、 DAC、アンテナが同じ LSI 上に搭載されるようになってきた。これによって、画像劣化、音声劣化、受信感度劣化といった、ノイズによる特性劣化が大きな問題になってきてる。原因は、デジタル回路のスイッチングが原因の電源ノイズがアナログマクロのノイズ耐性を超え、伝播するからである。ノイズは、チップ、パッケージ、ボードを介して、ノイズに敏感なアナログマクロに伝播する。特に、チップに於ける、伝播経路は、注意を行い、設計、検証を行う必要がある。なぜなら、他の伝播経路と比べ、インピーダンスが小さいからである。

ノイズ耐性を超えたノイズの伝播を防ぐために、デジタル回路とアナログマクロの電源、 GND は分離されている。しかし、デジタル、アナログ混載回路において、同じ Si 基板上にアナログマクロとデジタル回路が搭載されているので、電源ノイズは、 Si 基板を介してアナログマクロに伝播する（図 5.1 参照）。その電源ノイズはアナログマクロの特性を劣化させる。この基板を介して伝播する電源ノイズを基板ノイズを呼ぶ。

回路設計において、基板ノイズを減少させることは、重要である。デジタル回路とアナログマクロの距離を増加させると、インピーダンスが増加するので、基板ノイズは、減少する。しかし、この方法では、 LSI の面積は増加してしまう。

他の基板ノイズを低減する方法として、ガードリング手法が提案されている。アナログ、デジタル混載システムに於ける、ガードリングの効果は、 [1], [9], [35], [36], [37]に示されている。 p+ ガードリングは、最も一般的に用いられている方法である。ガードリングは、ノイズに敏感なアナログマクロの周りに配置される。ガードリングは、アナログマクロの GND か、ガードリング用専用 PAD に接続を行う必要がある。この PAD までの、接続配線は、非

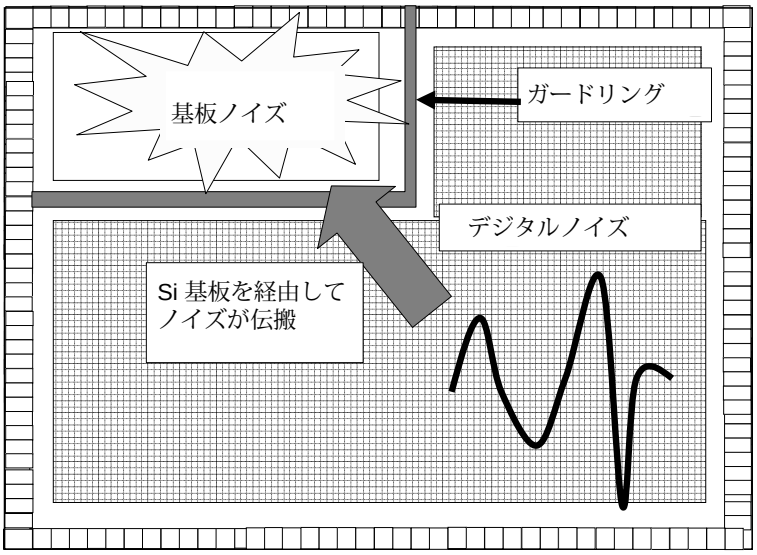

Figure 5.1: 従来手法 - ガードリング手法

常に低インピーダンスにする必要がある。しかし、パッケージのインダクタンスのため、低インピーダンスとし、ノイズをチップ外に逃すことは、難しい。この方法では、頻繁に、ノイズ低減効果が小さくなる。よって、 PAD 接続が不要な方法が提案されている。この方法では、ガードリングとして、 n-well 領域もしくは trench 領域が用いられる。これらの方法は、基板表面の低抵抗部分の抵抗を増加させることによって、分離を図る。これらの方法は、基板表面の抵抗を増加させることによって、ノイズに敏感なアナログマクロを分離する。これによって、電流は、抵抗の高い基板底面を流れることになる。基板表面の抵抗を上げることは、伝播ノイズを減少させる。しかし、基板表面の n-well 領域もしくは trench 領域で伝播ノイズを減少させる効果は、小さい。他の方法として、 triple well 構造を用いる方法がある。この構造は、高周波に対し効果がある。しかし、チップの面積は増加し、製造コストも増加するという問題がある。

我々は、基板ノイズのソースノード（基板ノイズを発生する原因のノード）の電圧を減少させる基板ノイズ削減アルゴリズムを提案する。提案アルゴリズムは、アナログマクロに最も電流を供給しているデジタル回路を特定する。そして、デジタル回路の GND ネットワークの配線幅を変更することによって、デジタル回路の電圧を改善し、アナログマクロに流れ込む電流量を減少させる。これらの処理は、基板ノイズ制約を満足するまで行われる。よって、提

案アルゴリズムは、基板ノイズ制約を満足しつつ LSI の面積を増加させないという点で、ガードリングを行う方法より優れている。

5.2 章では、電源 /GND/ 基板ネットワークモデルと電源 /GND/ 基板ネットワークモデルの抵抗メッシュモデルを提案する。 5.3 章では、基板ノイズ最適化問題の定式化を行う。 5.4 章では、基板ノイズ最適化アルゴリズムの概要、基板ノイズ最適化アルゴリズムの詳細を説明する。 5.5 章では、計算機実験結果について述べる。

5.2 基板ノイズの解析モデル

5.2.1 電源 /GND/ 基板ネットワークモデル

電源 /GND/ 基板ネットワークモデルは 2 章で述べたマクロモデリングアルゴリズムを用いて作成する。基板ノイズ最適化において、基板ノイズ制約は、電源 /GND/ 基板ネットワークモデルの誤差を加味し、計算される。よって、基板ノイズ最適化の結果は、ドメインにマクロモデリングアルゴリズムを用いたにもかかわらず基板ノイズ制約を満足している。よって、提案の基板ノイズ最適化アルゴリズムは、高速に精度良く解析を行うことができる。

図 5.2 は、電源 /GND/ 基板ネットワークモデルの例を示す。この例では、 1 つのアナログマクロが存在し、左下のノードが、アナログマクロを示している。デジタル電源 /GND ネットワークとアナログ電源 /GND ネットワークは、ノイズ伝播防止の観点から、分離されている。電源 /GND/ 基板ネットワークモデルにおいて、電源グリッドの各抵抗は、各層の合成抵抗となっている。電源グリッドの容量は、グリッド上の寄生容量とオンチップデカップリング容量の合成容量となっている。説明を簡単にするために、電源 /GND/ 基板のグリッドは、等間隔のグリッドで作成する。通常、基板ノイズ回路シミュレーションは、チップのインダクタンス値が小さく結果に大きな影響を与えないので無視する。そこで、我々も本論文では、チップ内のインダクタンスを考慮しない。

5.2.2 電源 /GND/ 基板ネットワークモデルの抵抗メッシュモデル

本章では、抵抗メッシュモデルについて説明を行う。このモデルは、基板ノイズのソースであるデジタルノード（抵抗メッシュモデルのデジタル領域のノード）を探索するために用いる。また、抵抗メッシュモデルは、基板ノイズソースのデジタルノードの電圧を改善するために、配線の幅を変更すべき電源 /GND ネットワークの枝を探索するために用いる。抵抗メッシュモデルは、基板ノイズ制約を満足していない解析時間の基板ノイズ回路シミュレーション

の結果から作成される。抵抗メッシュモデルの枝の抵抗は、枝を流れる電流とノードの電圧から計算される。また、抵抗メッシュモデルの枝の電流と抵抗メッシュモデルのノードの電圧は、基板ノイズ制約を満足していない解析時間の基板ノイズ回路シミュレーション結果から作成される。この時、パッケージのインダクタンスは抵抗として表現される。図 5.3 は、電源 /GND/ 基板ネットワークモデルの抵抗メッシュモデルの例である。以下の表記は、基板ノイズ最適化問題を定式化する時に用いる。基板ノイズ解析グラフ $= (N_u, E_b)$ は、電源 /GND/ 基板ネットワークモデルの抵抗メッシュモデルとして表現される。本論文で用いる記号を以下で定義する。

N_d: 電源 /GND/ 基板ネットワークのノードの集合
N_{pa}: 電源 /GND　PAD のノードの集合
N_p: 電源ネットワークのノードの集合
N_g: GND ネットワークのノードの集合
N_s: 基板ネットワークのノードの集合
$N_u = N_d \cup N_{pa}$
$N_d = N_s \cup N_g \cup N_p$
$n = |N_u|$
E_{pg}: N_p のノードと N_g のノードを結ぶ枝の集合
E_{gs}: N_g のノードと N_s のノードを結ぶ枝の集合
E_{papg}: N_g/N_p のノードと N_{pa} のノードを結ぶ枝の集合
E_p: N_p のノード間を結ぶ枝の集合
E_g: N_g のノード間を結ぶ枝の集合
E_s: N_s のノード間を結ぶ枝の集合
$E_b = E_{pg} \cup E_{gs} \cup E_p \cup E_g \cup E_s \cup E_{papg}$
$c_{i,j}$: 枝 $\{i, j\}$, $i, j \in N_u$ の電流
v_i: ノード $i, i \in N_u$ の電圧
$r_{i,j}$: 枝 $\{i, j\}$, $i, j \in N_u$ の抵抗
$g_{i,j}$: 枝 $\{i, j\}$, $i, j \in N_u$ のアドミッタンス
i_i: ノード $i, i \in N_u$ の電流源

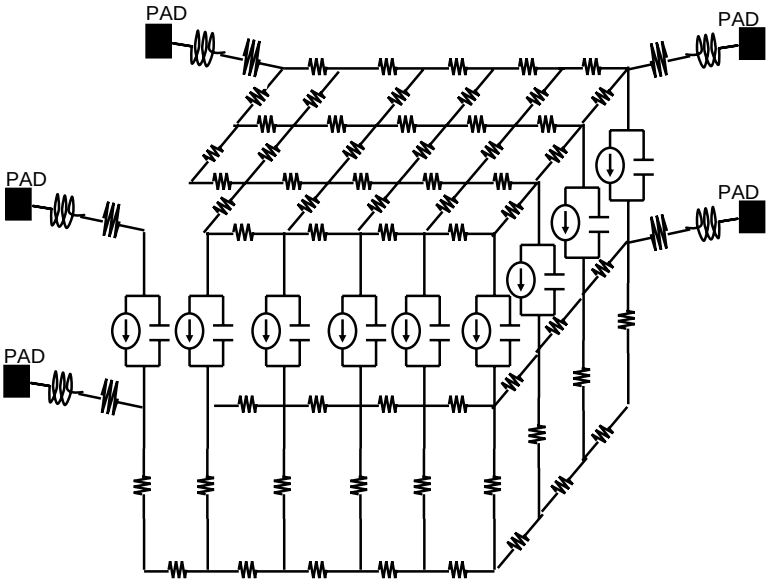

Figure 5.2: 電源 /GND/ 基板ネットワークモデルの例

5.3 基板ノイズ最適化問題の定式化

5.3.1 電圧ドロップ制約

デジタル回路の論理が、正確に伝播するには、電圧ドロップ制約を満足しなくてはならない。これは、 STA(Static Timing Analysis) システムから決まる設計マージンから決定される。V_{min} と V_{max} は電源ネットワークの最大電圧、最小電圧と定義する。よって、電圧ドロップ制約は以下の様に定義できる。

$$V_{min} \leqq v_i \in N_g \leqq V_{max} \tag{5.1}$$

5.3.2 回路制約

電圧の集合はノード電圧のベクター $V = \{v_1, v_2, , v_n\}^T$ として定義できる。電流の集合は、ノードの電流の集合 $I = \{i_1, i_2, , i_n\}^T$ として定義できる。キルヒホップの法則によって、回路方程式は作成される [25]。この回路制約は以下の様に表現できる。

$$A \times V = I, \tag{5.2}$$

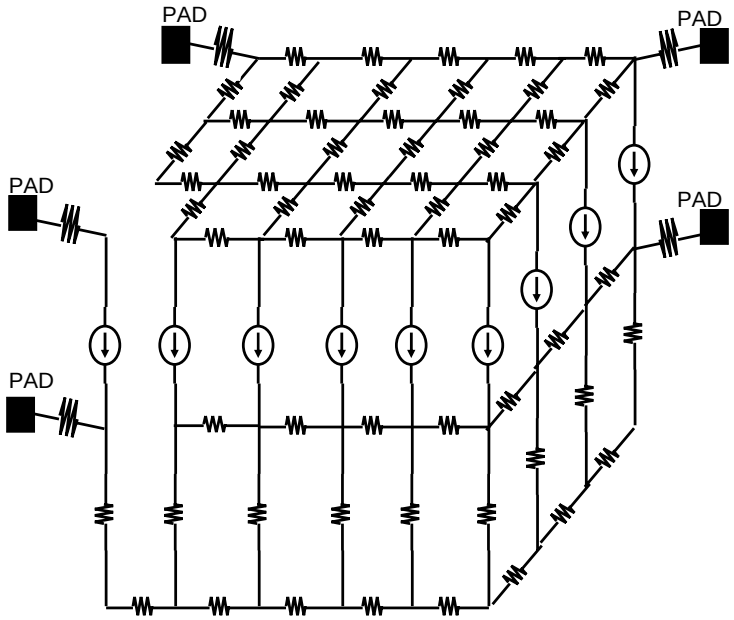

Figure 5.3: 抵抗メッシュモデルの例

ここで、 A は、 $n \times n$ アドミッタンス行列を表す。回路方程式の作成方法の詳細は [25, 26]に述べられている。

5.3.3 デザインルール制約

配線の最小幅、最大幅、配線の最小間隔、ビアの最小幅、最大幅、最小間隔といった LSI の設計ルールが定義されている。これらの設計ルールは必ず守らなければならない。加えて、配線に用いることができる面積にも制限がある。よって、アドミッタンスは最大と、最小の制約を持つ。G_{min} と G_{max} は、デザイン制約から決まる GND ネットワークの配線の最大アドミッタンス、最小アドミッタンスと定義する。よって、デザインルール制約は以下の様に定義できる。

$$G_{min} \leqq g_{i,j} \leqq G_{max} \tag{5.3}$$

5.3.4 基板ノイズ制約

アナログマクロが基板ノイズをノイズ耐性より多く受けた場合、アナログマクロは、特性劣化を起こす。このノイズ耐性を基板ノイズ制約と呼ぶ。このノイズ耐性を、$I_{noisetorelance}$ と定義する。基板を経由して、アナログノード i に伝播する電流量の総量を以下で定義する。

$$\sum_{j \in Ns} c_{i,j} \tag{5.4}$$

よって、基板ノイズ制約は以下で定義できる。

$$\sum_{j \in Ns} c_{i,j} \leqq I_{noisetolerance} \tag{5.5}$$

5.3.5 目的関数

電源 /GND ネットワークの配線面積を最小化することは、信号配線の配線性を高め、タイミングクロージャーを容易にし、製造コストを小さくする。よって、目的関数は、電源 /GND ネットワークの総配線面積最小とする。$l_{i,j}$ と $w_{i,j}$ は枝の配線の長さと幅と定義する。シート抵抗を $\rho(\Omega$ / square) とする。電源配線の総配線面積は以下で定義できる。

$$Area = \sum_{i,j \in N_g} l_{i,j} \times w_{i,j} \tag{5.6}$$

枝 $\{i, j\}$ の抵抗は以下で表すことができる。

$$r_{i,j} = \rho \times l_{i,j}/w_{i,j} \tag{5.7}$$

配線の長さは、チップのサイズで決定するので、一定であるため、$l_{i,j}$ として定義し、配線面積は $w_{i,j}$ を最小化すること、即ち総抵抗を最大化することとなる。アドミッタンスは抵抗の逆数であるため、配線面積はアドミッタンスを減少させることによって、最小化できることとなる。よって、目的関数は、以下で定義できる。

$$Obj : \min \sum_{i,j \in N_g} g_{i,j} \tag{5.8}$$

5.3.6 基板ノイズ最適化問題

以上の結果から、基板ノイズ最適化問題は以下で定式化できる。

$$\begin{aligned} Minimize \quad & \textstyle\sum_{i,j \in N_g} g_{i,j} \\ Subject \ \ to \quad & V_{min} \leqq v_i \leqq V_{max} \\ & G_{min} \leqq g_{i,j} \leqq G_{max} \\ & A \times V = I \\ & \textstyle\sum_{j \in Ns} c_{i,j} \leqq I_{noisetolerance} \end{aligned} \tag{5.9}$$

本論文では、アナログマクロに最も電流を流しているデジタルノードを探し、電源ネットワークの配線面積の増加を抑えて、そのデジタルノードの電圧を減少させることによって解く事ができるアルゴリズムを提案する。

5.4 基板ノイズ最適化アルゴリズム

5.4.1 フロー概要

基板ノイズの原因は、デジタル回路がスイッチングする時に生じる電源ノイズである。よって、提案アルゴリズムでは、この電源ノイズを減少させることによって、基板ノイズを減少させる。電源ノイズを減少させるために、提案アルゴリズムでは、 GND 配線の総面積の増加を最小化するように GND ネットワークの枝の幅を、変更する。

LSI 上には、多くのデジタル回路が搭載されている。このデジタル回路は、アナログマクロに異なった影響を与える。 GND 配線の総面積の増加を抑えるためには、最もアナログマクロに影響を与えるデジタル回路を見つけ、その回路の電圧を改善させるべきである。よって、提案アルゴリズムは以下の 2 つのフェーズから成り立っている。

phase1 アナログマクロに最も影響を与えるデジタルノード（デジタル回路）を探す。

phase2 そのデジタルノードの電圧を改善する。

提案アルゴリズムは、基板ノイズ制約違反がなくなるまで、この 2 つのフェーズを繰り返す。フェーズ 1 では、デジタル回路からアナログマクロへ流れる電流が、アナログマクロに流れ込む電流経路を探索することによって求められる。最もアナログマクロに電流を流しているデジタルノードは、最もアナログマクロに影響を与える。よってそのノードの電圧を改善す

る。フェーズ 2 では、まず、デジタルノードの電圧を決定にしているパスの中で最も影響のあるパスを求める。次に、最も影響のあるパス上で最も影響のある枝を求め、その枝の幅を変更する。以下に提案アルゴリズムのアウトラインを示す。

電源 /GND/ 基板ネットワークモデルは、マクロモデリングアルゴリズムを用いて作成される。
基板ノイズ解析（回路シミュレーション）が実行される。
While（基板ノイズ制約を満足しない解析時間がある。）
{
基板ノイズ制約を満足しない解析時間に対して、抵抗メッシュモデルを作成する。
While(基板ノイズ制約を満足しない。){
抵抗メッシュモデルを用いて最もアナログマクロに影響のあるデジタルノード
を探す。
そのデジタルノードの電圧決定に、最も影響のあるパスを求め、そのパス上で
デジタルノードの電圧決定に最も影響のある枝を求める。そして、その枝の幅を
太くする。
抵抗メッシュモデルの回路シミュレーションを行う。
}
基板ノイズ解析を行う。
}

以降の章で、基板ノイズソース特定アルゴリズム（フェーズ 1）と電圧ドロップ改善アルゴリズム（フェーズ 2）の詳細アルゴリズムについて説明を行う。

5.4.2 基板ノイズソース特定アルゴリズム

本章では、基板ノイズソース特定アルゴリズムを説明する。このアルゴリズムは、アナログマクロに最も電流を多く伝播させ、アナログマクロの特性を悪化させるデジタルノードを特定する。ノイズに敏感なアナログマクロの特性劣化は、アナログマクロに伝播した電流によって引き起こされる。そこで、各デジタルノードからアナログマクロに流れる電流量を求める。最も電流を多く伝播させているデジタルノードが基板ノイズソースとなる。基板ノイズソース特定アルゴリズムを以下に示す。

STEP 1: デジタルノードからアナログマクロへの電流経路（パス）を求める。

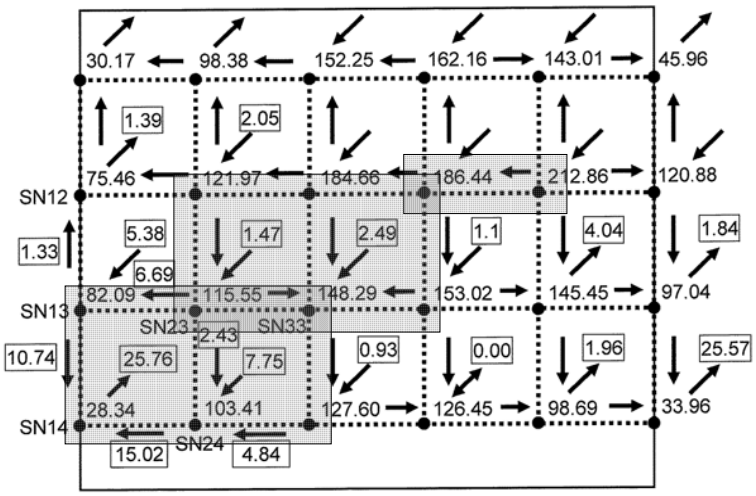

Figure 5.4: 抵抗メッシュモデルの基板ネットワーク部分のノードの電圧と枝の電流フロー

STEP 2: 基板ノイズソースの候補ノードの特定を行う。

STEP 3: 各候補ノードからアナログマクロへの伝播電流量を求める。

STEP:4 候補ノードから基板ノイズソースノードを選択する。

STEP 1 では、デジタルノードからアナログマクロへの伝播経路を特定する。電流は、電圧の高いほうから低い方へ流れる。電流の流れる経路がパスである。よって、アナログマクロから、デジタルノードへの電流の流れを探索することによって、パスを求める。アナログマクロを探索の初期点として、電源 /GND/ 基板ネットワークモデルの抵抗メッシュモデル上を GND ネットワークのノードに到達するまで探索を行う。図 5.4 は、図 5.3 の抵抗メッシュモデルの基板ネットワークのノードの電圧と枝の電流フローを示す。ノードの横の数字は、そのノードの電圧（mV）を表す。各枝に沿った縦、横の矢印は、電流の流れる方向を示す。斜めの矢印は、 GND ノードと基板ノード間を流れる電流の流れる方向を示す。また、矢印につけられた箱の中の数字は、電流 (mA) を示す。図 5.4 では、いくつかのシンボルと数字が掲載されていない。図 5.4 で、網掛け領域上のパスは、電流経路として選択されたパスである。

STEP 2 では、基板ノイズソースの候補ノードを特定する。アナログマクロに電流を供給するデジタルノードが、候補のノードである。選択されたパス上のノードでデジタル GND ネットワークの GND ノードが候補ノードとなる。よって、パス探索時、パスの終点のノード

を求め、基板ノイズのソースノードの候補とする。パス上のノード（終点ではないノード）は、電圧勾配の中点であるため、基板ノイズソース候補ノードとしては、望ましくない。電圧ドロップ改善アルゴリズムによって、電圧勾配の中点にあるノードの電圧が改善されると、電圧傾斜がより大きくなり、アナログマクロに流れる電流が増加する結果となる。よって、パスの終点のノードを基板ノイズソースノードの候補として選択する。

STEP 3 では、アナログマクロへの伝播電流量が、各基板ノイズソースノードの候補ノードから、アナログマクロへの経路を探索することによって算出される。

SETP 4 では、基板ノイズソースノードを、基板ノイズソースノードの候補ノードの中から選択する。アナログマクロに最も多くの電流を伝播している候補ノードが改善されるべきノードである。よって、電流を最も多くアナログマクロで伝播している候補ノードが、基板ノイズソースノードとして選択される。

5.4.3 電圧ドロップ改善アルゴリズム

本章では、配線面積の増加を最小に押さえながら、基板ノイズソースノードの電圧を改善する電圧ドロップ改善アルゴリズムについて述べる。電源 /GND ネットワークの配線面積の増加を抑え、電圧を改善するには、基板ノイズソースノードの電圧を改善するのに最も効果のある配線の幅を変更すべきである。よって、提案の電圧ドロップ改善アルゴリズムでは、基板ノイズソースノードの電圧に最も影響のある配線を探し、その配線の幅を変更する。

以下に提案の電圧ドロップ改善アルゴリズムのアウトラインを示す。

STEP1: 基板ノイズソースノードの電圧を決定するパスを選択する。

STEP2: そのパスの中で、最も基板ノイズソースノードの電圧決定に影響のあるパスを 1 つ見つける。

STEP3: パス上の枝で、 最も基板ノイズソースノードの電圧決定に影響のある枝を 1 つ見つける。

STEP4: 枝の幅を更新する。

以下では、 STEP1、 STEP2、 STEP3、 STEP4 のアルゴリズムの詳細を説明する。

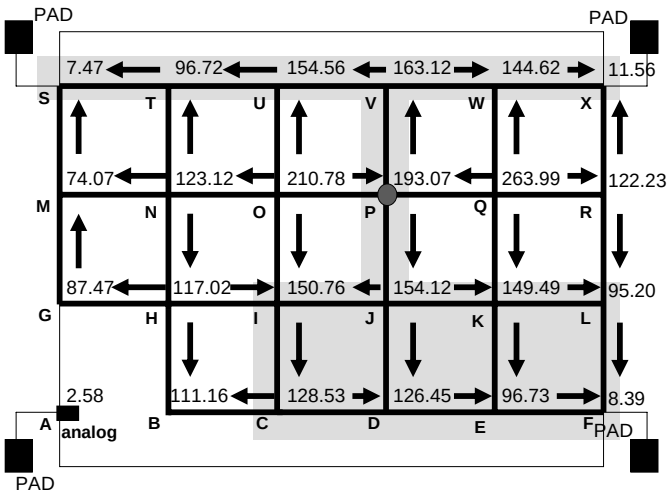

Figure 5.5: 抵抗メッシュモデルの GND ネットワーク部分のノードの電圧と枝の電流フロー

5.4.4 オーバーフローパスの抽出

基板ノイズソースノードの電圧を改善するには、改善の効果があるパスを選択することが重要である。ここでは、電圧を改善する効果のあるパスを如何に選択するかについて述べる。ノードの電圧ドロップは、ある基準のノード（たとえば PAD）からそのノードまでのパスの抵抗と電流で決まる。電流は、電圧の高いノードから低いノードに流れる。電流の経路はパスである。よって、基板ノイズソースノードから、電流フローを探索することによって、パスを求める。オーバーフローパスは、基準ノード（PAD）から基板ノイズソースノードへのパスと定義する。

図 5.5 に図 5.3 の抵抗メッシュモデルの GND ネットワーク部分のノードの電圧と枝の電流フローを示す。図 5.5 で、網掛け領域上のパスは、オーバーフローパスとして選択されたパスを示す。このパスは、基板ノイズソースノードの電圧を決定している。図 5.5 で、$\{P, V, W, X\}$, $\{P, V, U, T, S\}$, $\{P, J, K, L, F\}$, $\{P, J, K, E, F\}$, $\{P, J, D, E, F\}$, $\{P, J, I, C, D, E, F\}$, がオーバーフローパスを示す。これらのパスは、基板ノイズソースノードへの電流経路を示す。オーバーフローパス上のすべての枝は、基板ノイズソースノードの電圧を改善する効果がある。オーバーフローパスを抽出することは、提案アルゴリズムに於いて、解の探索経路を減少させる点で重要である。

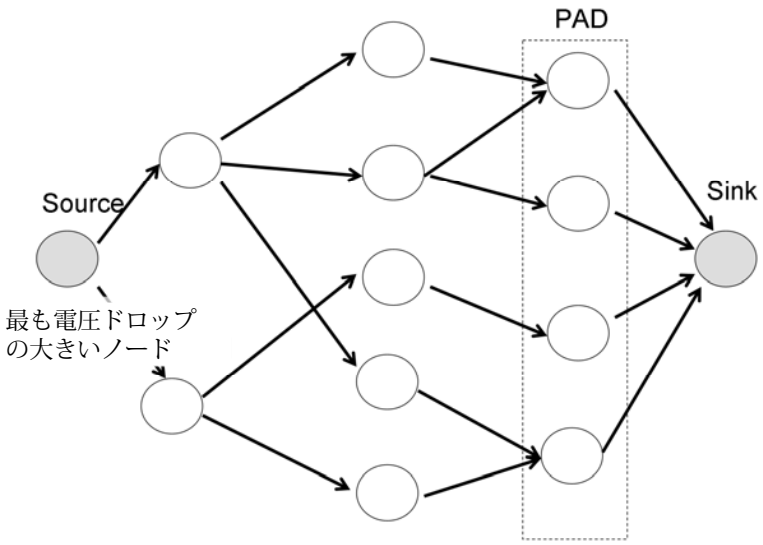

Figure 5.6: 電流フロー有向グラフの例

基板ノイズソースノードから PAD への電流の流れを表す、電流フロー有向グラフ作成アルゴリズムについて述べる。電流フロー有向グラフ $= (V, E)$ は、基板ノイズソースノードから、 PAD へのパスを抵抗メッシュモデルを探索しならが作成される。基板ノイズソースノードは、電流フロー有向グラフ $= (V, E)$ のソースノード $\mathrm{s} \in \mathrm{V}$ とする。ソースノードから、PAD ノードまで、電圧が高くなる方向に深さ優先探索（幅優先探索） [27] を実行し、同時に、電流フロー有向グラフの有向枝 $\{i, j\} \in \mathrm{E}$ を作成する。 PAD に到達すると、探索は終了し、電流フロー有向グラフのシンクノードが作成され、 PAD ノードからシンクノードへの有向枝を作成する。電流フロー有向グラフの枝の方向は、電流の流れと逆となる。図 5.6 に電流フロー有向グラフの例を示す。

5.4.5 クリティカルパスの抽出

枝の両端の電圧の差が大きい枝がノードの電圧ドロップに大きな影響を与える。まず最初にボトルネック枝を定義する。枝の右のノードの電圧と左のノードの電圧を v_i、 v_j と定義する。以下の条件を満足する枝をボトルネック枝と呼ぶこととする。

$$|v_i - v_j| \geqq \text{ボトルネック制限値} \tag{5.10}$$

ボトルネック制限値は、電圧ドロップ制約から定義する。ボトルネック枝は、電圧を改善するために、枝の抵抗を変更すべき枝を示す。加えて、ボトルネック枝は、フロー問題において、枝の容量をボトルネック制約から決定した場合、枝の容量を超えてフローが流れている枝を示す。オーバーフローパス上のボトルネック枝は、最も電圧改善効果のある枝である。

次に、オーバーフローパスから改善を行うべきパスをどの様に選択するかについて述べる。クリティカルパスは、基板ノイズソースノードの電圧を改善するのに最も効果があるパスと定義する。最も多くのボトルネック枝を持つパスが、クリティカルパスである。よって、電流フロー有向グラフのボトルネック枝に小さなコストを与え、それ以外の枝には大きなコストを与え、最小コストのパスを見つける。このパスがクリティカルパスとなる。

クリティカルパスを探す方法の詳細について述べる。電流フロー有向グラフの枝は、基板ノイズソースノードの電圧を改善する候補枝である。枝に容量を超えてフローが流れている枝には小さなコストを、それ以外の枝、容量を超えてフローがながれていない枝には大きなコストを与える。コスト関数の例を以下に示す。

$$cost_{i,j} = B^{(flowi_{i,j} - capacity_{i,j})} \quad 0 < B < 1 \tag{5.11}$$

ここで、$cost_{i,j}$ は、枝 $\{i,j\}$ のコスト、B は定数、$flow_{i,j}$ は、枝 $\{i,j\}$ のフロー、$capacity_{i,j}$ は、枝 $\{i,j\}$ の容量を示す。ソースノードからシンクノードへのパスで、最もコストの和が小さいパスが、最も改善効果のあるパスとなる。なぜなら、容量を超えてフローが流れているためである。そこで、ダイクストラアルゴリズム [27]を用いて最小のコストを持つパスを見つける。このパスが、クリティカルパスとなる。

5.4.6 クリティカル枝の抽出

クリティカルパス上のボトルネック枝を改善するアルゴリズムについて述べる。クリティカルパス上の枝で最もコストが小さい枝が最も電圧改善に効果のある枝である。よって、クリティカルパス上の最もコストの小さい枝を更新する。クリティカル枝を、クリティカルパス上の枝で最もコストの小さい枝と定義する。フロー問題として考えた場合、パス上の最もコストの小さい枝が、フローの妨げとなっているので、最も小さいコストを持つ枝の容量を増加させる。

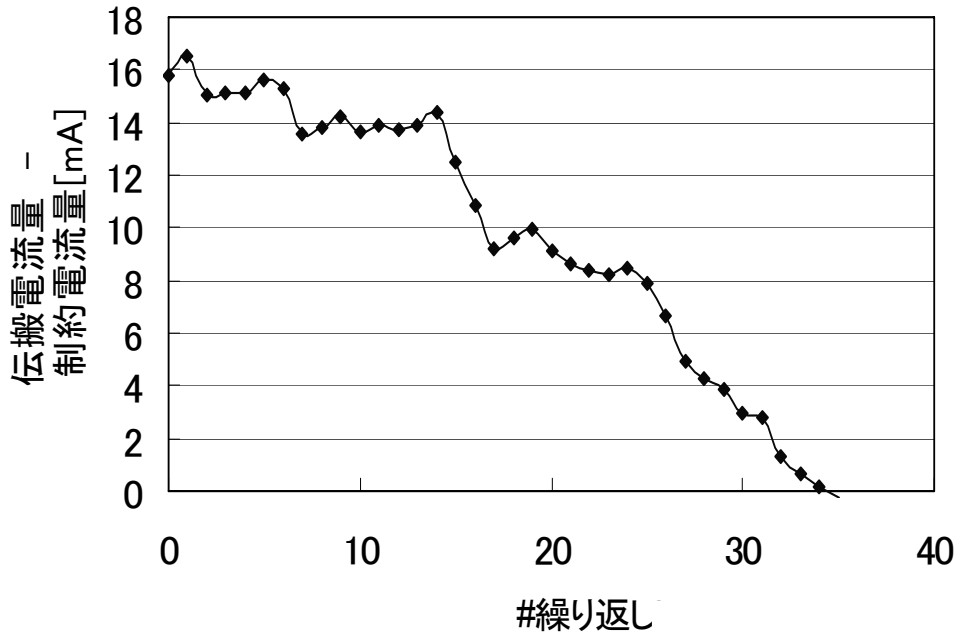

Figure 5.7: 伝播ノイズ量削減過程

次に、基板ノイズソースノードの電圧を改善するのに最も効果のあるオーバーフローパス上の枝は、ネットワークフローとして考えた場合、どの様な意味があるかについて述べる。問題 $A \times V = I$ は、枝を流れる電流を求めることによって、ノードの電圧を求める問題と考えることができる。枝の右のノードと左のノードの電圧差が大きい場合、枝の抵抗は、枝を流れる電流に対し大きいと考えることができる。即ち、容量に対し、フローが大きい。提案のアルゴリズムでは、容量を超えてフローが流れている枝を捜し、容量を増加させる。即ち枝の幅を増やし、抵抗を小さくする。結果電圧が改善する。

5.5 計算機実験結果

2.6GHz CPU、 4GB メモリのリナックスマシンを用いて、計算機実験を行った。 1 億個のデジタルプリミティブブロックと 1 つのアナログマクロを持つデータで実験を行った。図 5.7 は伝播電流量の改善過程を示した図である。図 5.7 において、 x 軸は、 STEP2 から STEP4 の繰り返し回数を示す。また、 y 軸は、基板ノイズ制約と伝播電流量の差を示す。 37 回の繰り返しで、 LSI の面積を増加させる事なしに基板ノイズ制約を満足する結果を得ることができた。最適化にかかった時間は 4m6s と、実用的な時間であった。

5.6 まとめ

電流フローパスを探索することによって基板ノイズを減少させるアルゴリズムを開発した。提案アルゴリズムは、アナログマクロに最も電流を伝播させているデジタルノードを特定する。そのノードの電圧を改善することによって、アナログマクロに伝播する電流量を減少させる。よって、 LSI の面積を増加させることなしに、基板ノイズ制約を満足することができる。計算機実験結果から、 LSI の面積増加なしに、基板ノイズ制約を満足する結果を得ることができることを確認した。

Chapter 6

結論

近年、プロセス技術の進歩に伴い、回路の動作速度は向上した。また、搭載されるデジタル回路規模が増加し、これに伴い大電流化が進み、デジタル回路におけるノイズ、即ち電圧降下（電圧ドロップ）は増大の傾向にある。一方、低消費電力化の要求によって、低電圧化が進んでいる。このため、相対的にノイズが大きくなり、デジタル回路のノイズ耐性は弱くなる傾向にある。デジタル回路に於いては、電圧ドロップによる誤動作や電圧ドロップによる遅延劣化が原因の動作速度低下が問題となっている。これは、高速化要求への障害となっている。

一方、ノイズに敏感なアナログマクロ（HDMI、 DDR、 USB 等高速 IO マクロ、 ADC、DAC マクロ、アンテナマクロ）とデジタル回路が 1 チップに搭載されるようになってきた。これによって、アナログマクロの誤動作や特性劣化が顕著化してきている。ノイズ問題は、一旦発生すると解決に多くの時間を要することが多く、大きな問題となっている。回路が放出したノイズをその回路自身が受け誤動作を引き起こす自家中毒は、 LSI を製造するメーカにとっては、最も考慮すべきノイズである。また、近年、 EMC では、 VCCI 等の規格が制定され、規格を満足しないシステムの販売が禁止されている。さらに、ノイズイミュニティ (ノイズ耐性) に対する要求も増大している。加えて、近年の価格競争の激化によって、低コスト化の要求が増大している。 LSI のサイズは、コストに直結するため、コスト削減に於いて、最も重要な項目である。このため、チップ内で大きな面積を占めている、電源 /GND ネットワークやオンチップデカップリング容量削減は重要な項目である。また、設計期間の短縮要求によって、高速に、精度良く解析、設計を行える設計手法が重要となってきている。これらの要因から、正確な電圧ドロップ、基板ノイズの解析モデルの構築が重要となってきている。モデル化に於いて、以下の 4 つが重要な項目である。

１） ノイズ（電圧ドロップ）の発生を正しくモデル化する

２） 電圧ドロップによる遅延変動を正しくモデル化する

３） 伝播経路を正しくモデル化する

４） ノイズを受けた場合のアナログマクロの動作を正しくモデル化する

そこで、本論文では、これらの 4 つを正確にモデルし解析を行う方法、即ち、マクロモデリングによるノイズ解析アルゴリズムを提案する。本アルゴリズムは、非常に高速であるため、最適化アルゴリズムのノイズ耐性チェックとして有効である。また、コスト削減のために、ノイズ耐性を満足し、かつ電源 /GND ネットワークの総配線面積、オンチップデカップリング容量の総面積を最小化することが重要である。加えて、アナログマクロに伝播する基板ノイズをチップサイズ、配線層数の増加なしに、アナログマクロのノイズ耐性以内となる様に減少させることも重要である。これらを実現するために、本論文では、上で述べたマクロモデリングによるノイズ解析アルゴリズムを適用し、高速に高精度に電源 /GND ネットワークの最適化設計を行う以下の 3 つのアルゴリズム：

１） 電源 /GND ネットワーク最適化アルゴリズム

２） オンチップデカップリング容量最適化アルゴリズム

３） 基板ノイズ最適化アルゴリズム

の提案を行う。

第 2 章では，大規模回路に対して、電圧ドロップ、基板ノイズ解析を高速に、高精度に行う方法について議論する。デジタル回路のタイミングマージンを悪化させ、アナログマクロの特性を劣化させる電圧ドロップ、基板ノイズを高速に、高精度に解析することは、回路が正しく動作するかどうかを判断するために重要な課題である。 Badaroglu は、プリミティブブロックレベルでマクロモデリングを行い、電圧ドロップ、基板ノイズ解析を行う方法を提案した (IEEE trans. 2006)。この方法は、トランジスタレベルで解析を行うのではなく、プリミティブブロックレベルでモデル化を行い、解析を行うため、トランジスタレベルの解析アルゴリズムと比べ高速化は期待できる。しかし、 SoC の様な大規模回路に於いては、処理時間、メモリサイズの面で適用することは困難である。そこで、本アルゴリズムは、大規模回路を高速に、精度良く解析する為に、 LSI を複数のドメインに分割し、そのドメイン毎にマクロモデルを作成し、電圧ドロップ解析、基板ノイズ解析を行う。ドメイン数を削減することによってノード数が減少するので、計算複雑度は大きく減少する。しかし、解析精度は悪化する。そこ

で、ドメインサイズ、解析時間、解析精度のトレードオフ関係を考察した。考察結果から、ドメインの電流密度変動係数と見積もり基板ノイズ誤差の間に線形の関係があることが判明した。この結果から、ドメインサイズをドメインの電流密度変動係数を元に決定することによって、精度と処理時間の両方を満足できることが判明した。提案アルゴリズムでは、この線形関係を利用し、まず目標誤差を設定し、その誤差に対応する電流密度変動係数を求め、各ドメインを電流密度変数係数以内となるようにドメイン分割を行う。これによって、精度と解析時間の両立を実現する。実験結果から、ノイズ誤差を 15 % に設定し、ドメイン分割を行った場合、見積もりジッタ（クロック周期のゆらぎ）の精度は 85% 以上であった。これは、電流密度変動係数による、ドメイン分割が正しく動作していることを示す。また、提案アルゴリズムは Badaroglu のアルゴリズムと比べ、 10000 倍以上高速であった。提案アルゴリズムを適用すると解析速度と精度の両方を満足できるので、数千億のトランジスタを持つマイクロプロセッサの様なシステム LSI に対しても適用可能な方法である。また、ノイズ耐性のチェックアルゴリズムとして最適化アルゴリズムで適用することが可能な有効な方法である。よって、提案アルゴリズムが効果的であることが結論づけられる。

第 3 章では、電圧ドロップ制約を満足し、同時に電源 /GND ネットワーク配線の総面積を最小化するアルゴリズムについて議論する。電源 /GND ネットワーク配線の総面積の増加は、配線性、タイミングクロージャーに大きな影響を与える。配線層数を最小化し、製造コスト、製造 TAT を削減することが、コスト削減の目的から求められている。 Zhao は、この問題を数学的に定式化し、非線形計画問題として解く方法を提案している (DAC 2007)。しかしこの方法では、 SoC の様な多くの素子を持つ問題に適用することは難しい。また、緩和のため、通常解が悪化する。一方、 Singh は、ノイズ制約を満足しない場合、ドメインを満足しないドメインの配線幅を均一に増加させる欲張り最適化アルゴリズム（greedy optimization algorithms）を提案した (IEEE trans. 2006)。このアルゴリズムの 1 つの欠点は、均一に電源 /GND ネットワークの配線の抵抗を変更するので、電源 /GND ネットワークの配線面積が大きくなることである。提案アルゴリズムは、非線形計画問題を逐次線形計画法で解くのではなく、電流をフロー、許容電圧ドロップ値を容量として捕らえ、ネットワークフロー問題として解くことによって解を求める。即ち、ネットワークの枝の容量を増やす既存ネットワークの最小費用改良問題の拡張問題として捉え解を求める。これによって、非線形計画問題を解く時に用いる緩和を行う必要がなく、良い解を得ることができる。提案アルゴリズムは、まず、電圧ドロップに影響のあるパスを求める。これは、解析空間を絞り、解析時間の向上に有効であ

る。次に、そのパスの中でフローが容量を最も超えて流れているパスを 1 つ求める。その後、パス上の枝で電圧ドロップに最も影響のある枝を見つけ、その枝の抵抗を変更、即ち容量を増加させることによって電圧を改善する。提案アルゴリズムは電圧ドロップに最も影響のある枝を更新するため効率的な方法である。また、提案アルゴリズムは電流モードではなく電圧モードで制約を設定する方式を採用した。これによって、電圧ドロップ、即ち電圧を直接制約として扱うことが出来るようになり、 Zhao のアルゴリズムと比べ抵抗による電圧誤差を解消でき、良い解を得ることが出来るようになった。計算機実験結果から、 Singh のアルゴリズムと比べ電源 /GND ネットワークの配線面積を 1/3 に縮小することを示した。また、提案アルゴリズムは Singh、 Zhao のアルゴリズムでは電圧ドロップ制約を満足することができなかったデータに対しても電圧ドロップ制約を満足する解を得ることができた。よって、提案アルゴリズムが効果的であることが結論づけられる。

第 4 章では、電圧ドロップ制約を満足すると同時に、オンチップデカップリング容量の総面積を最小化するアルゴリズムについて議論する。オンチップデカップリング容量の追加は、電圧ドロップを解決する重要な手法である。オンチップデカップリング容量の追加は、チップ面積の増加、リーク電流の増加を招くため、追加する量を如何に減少させるかが重要な課題である。 Zhao は、この問題を数学的に定式化し、非線形計画問題として解く方法を提案している (DAC 2006)。しかしこの方法では、 SoC の様な多くの素子を持つ問題に適用することは難しい。また、緩和のため、解が悪化する。提案アルゴリズムは、非線形計画問題を逐次線形計画法で解くのではなく、電流をフロー、許容電圧ドロップ値を容量として捕らえ、ネットワークフロー問題として解くことによって解を求める。即ち、ネットワークの枝を追加する既存ネットワークの最小費用改良問題の拡張問題として捉え解を求める。非線形計画問題を解く時に用いられる緩和を行う必要がなく、良い解を得ることができる。提案アルゴリズムは、電圧ドロップに影響のあるパスを求め、そのパスの中でフローが容量を最も超えて流れているパスを 1 つ求める。次に、パス上のノードで、電圧ドロップに最も影響のあるノードを見つけ、そのノードにオンチップデカップリング容量を追加、即ちそのノードにソースノードからの枝を追加し、電圧改善を行う。提案アルゴリズムは、電圧ドロップに最も影響のあるノードにオンチップデカップリング容量を追加するため、効率的なアルゴリズムである。また、オンチップデカップリング容量から電荷が供給される時間を考慮し、電流が必要なノードと電流を供給するオンチップデカップリング容量の距離を、一定距離以内にする制約を設定し、実効的なオンチップデカップリング容量追加を可能にした。これによって効果のない箇所へのオンチップデカップリング

容量の追加を避けることができ、オンチップデカップリング容量の総面積の増加を抑えることができた。計算機実験結果から、提案アルゴリズムは実サイズデータに対し、処理時間が数分と実用的な方法である。また、 Zhao のアルゴリズムと比べ、オンチップデカップリング容量の総面積を 40 〜 50% に減らせることを示した。よって、提案アルゴリズムが効果的であることが結論づけられる。

第 5 章では、基板ノイズ削減を、デジタル電源 /GND ネットワークを最適化することによって実現するアルゴリズムについて議論する。デジタル、アナログ混載 LSI に於いて、デジタル電源ノイズは、 Si 基板を介してアナログマクロに伝播しアナログマクロの特性を劣化させる。この電源ノイズを、基板ノイズと呼ぶ。アナログマクロの特性劣化を防止するためには、基板ノイズの伝搬量をアナログマクロのノイズ耐性以内にすることが重要である。この問題を解決するために、ガードリング手法（Bronckers, IEEE trans. 2010）が提案されている。ガードリングを用いた従来手法では、ガードリング追加によって、 LSI の面積、製造マスク数が増加し製造コストが増加する。そこで、基板ノイズの原因がデジタル電源ノイズであることに着目し、デジタル電源ノイズ (ノイズ源) を直接削減し、基板ノイズの低減を図るアルゴリズムを提案する。直接デジタル電源ノイズを削減するため、 LSI の面積増加、製造マスク数増加無しに、基板ノイズを削減することができ、有効なアルゴリズムである。提案アルゴリズムは、アナログマクロに最も電流を伝播しているデジタル回路を電流経路探索によって特定する。次に、この最も電流をアナログマクロに供給しているデジタル回路の電圧ドロップを、 3 章で述べた電圧ドロップ改善アルゴリズムを適用し改善を行い、アナログマクロに伝播する電流の総量を削減する。提案アルゴリズムは、ガードリングを用いる従来手法と比べ、 LSI 面積の増加、製造マスク数の増加無しに、基板ノイズを低減できる点で優れている。計算機実験結果から、提案アルゴリズムは、実データサイズデータに対し、処理時間 4 分 6 秒で LSI 面積の増加無しに、 ADC のノイズ耐性以内に基板ノイズを削減する結果を得た。よって、提案アルゴリズムが効果的であることが結論づけられる。

高性能な LSI を如何に少ないコストで製造するかは、 LSI メーカーにとって永遠のテーマである。この問題は、ノイズとの戦いということもできる。 PI（Power Integrity）、 SI（Signal Integrity）、マイクロノイズ（Micro Noise）は、チップのみの特性で決まるものではなく、パッケージ、ボードの特性から大きな影響を受ける。特にボードは顧客設計であるため、個々の顧客によって特性が異なり、問題を複雑化させる。この様な状況の元、チップ / パッケージ / ボードをトータルに考え、ノイズ特性を満足し、チップ性能を十分に引き出し、

コストを低減することが今後も重要となると思われる。 DDR 等の高速 IO マクロが、多数搭載される様になり、チップのサイズが、 IO　 PAD の個数で決まる PAD ネックとなる品種が増加してきている。チップ内の微細化に比べ、 PAD を縮小することは難しく、今後この様な PAD ネックの品種は増加すると考えられる。本論文で提案したアルゴリズムは、動作を保障しコストを削減するために、重要なアルゴリズムとなると考える。また、コストの低減を図るには、チップサイズ、 PAD 位置、マクロ位置等は、設計の初期で決定を行う必要がある。実際に設計が始まってからでは、変更の自由度は小さく、十分な最適化が行えないためである。そこで、設計初期で如何にチップ性能、ノイズ特性等を見積もっていくのかが今後の課題である。

謝辞

本論文をまとめるにあたりまして，日頃から適切なご指導，ご助言を賜わります本学柳澤政生教授、後藤敏教授、松山泰男教授ならびに戸川望教授に深く感謝いたします。また，本機会を与えてくださいましたルネサスエレクトロニクス株式会社石井弘二副事業本部長、石川豊部長、大島泉課長に深く感謝いたします。ご多忙中ご指導を賜わりました本学大附辰夫教授、吉村猛教授に感謝いたします。

第 3 章の内容に関しまして，共同で研究に取り組んで頂きました梶田幹浩氏をはじめとする NEC 株式会社の関係者の方々、中元敏氏をはじめとするルネサスエレクトロニクス株式会社の関係者の方々に感謝いたします。

最後になりましたが，大学生活を過ごすにあたり、多岐にわたり協力してくれた主人と息子に感謝いたします。

Bibliography

[1] A. Afzali-Kusha, M. Nagata, N. K. Verghese, and D. J. Allstot, "Substrate Noise Coupling in SoC Design: Modeling, Avoidance, and Validation," Proc. of the IEEE, Vol. 94, No.12, pp. 2109-2138, December 2006.

[2] M. Badaroglu et al., "SWAN:High-Level Simulation Methodology for Digital Substrate Noise Generation," IEEE Trans. VLSI Systems, Vol. 14, No. 1, pp. 23-33, January 2006.

[3] M. van Heijningen, M. Badaroglu, S. Donnay, M. Engels, and I. Bolsens, "High-Level Simulation of Substrate Noise Generation Including Power Supply Noise Coupling," Proceedings of the 37th Design Automation Conference, pp. 446-451, 2000.

[4] K. A. Jenkins, W. Rhee, J. Liobe, and H. Ainspan, "Experimental Analysis of the Effect of Substrate Noise on PLL Performance," Topical Meeting on Silicon Monolithic Integrated Circuits in RF Systems, Digest of Papers, pp. 54-57, Jan. 2006.

[5] J. Kim, H. Shin, J. Lee, N. Moon, and J. Kong, "SilcVerify:An Efficient Substrate Coupling Noise Simulation Tool for High-Speed & Nano-Scaled Memory Design," Proceedings of the 8th International Symposium on Quality Electronic Design (ISQED ' 07), pp. 475-480, March 2007.

[6] R. Murgai, S. M. Reddy, T. Miyoshi, T. Horie, and M. B. Tahoori, "Sensitivity-based Modeling and Methodology for Full-chip Substrate Noise Analysis," Proceedings of the Design Automation and Test in Europe Conference and Exhibition Designers' Forum (DATE ' 04), Vol. 1, pp. 610-615, Feb. 2004.

[7] M. Nagata, "On-Chip Measurements Complementary to Design Flow for Integrity in SoCs," Proceedings of the 44th Design Automation Conference, pp. 400-403, 2007.

[8] G. Van der Plas et al., "High-level Simulation of Substrate Noise in High-Ohmic Substrates with Interconnect and Supply Effects," Proceedings of the 41st Design Automation Conference, pp. 854-859, 2004.

[9] D. K. Su et al., "Experimental Results and Modeling Techniques for Substrate Noise in Mixed-Signal Integrated Circuits," IEEE J. Solid-State Circuits, Vol. 28, No. 4, pp. 420-430, Apr. 1993.

[10] www.synopsys.com

[11] Y. Murasaka, M. Nagata, T. Ohmoto, A. Iwata, "Chip-level Substrate Noise Analysis with Network Reduction by Fundamental Matrix Computation," Quality Electronic Design, 2001 International Symposium on, pp. 482-487, 2001.

[12] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-meimand, "Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuits," Proceedings of the IEEE, Vol. 91, No. 2, pp.305-327, February 2003.

[13] Brett Peterson et al., "Automated Extraction of Model Parameters for Noise Coupling Analysis in Silicon Substrates," IEEE 2007 Custom Integrated Circuits Conference, pp. 853-856, 2007. 14

[14] Snakar P Debnath et al., "Calibration Based Methods for Substrate Modeling and Noise Analysis for Mixed-Signal SoCs," 20th International Conference on VLSI Design, pp. 887-892, 2007.

[15] T. Mitsuhashi and E. S. Kuh, "Power and Ground Network Topology Optimization for Cell Based VLSIs," Proceedings of the 29th Design Automation Conference, pp. 524-529, 1992.

[16] M. Zhao, R. Panda, B. Reschke, Y. Fu, T. Mewett, S. Chandrasekaran, S. Sundareswaran and S. Yan, "On-Chip Decoupling Capacitance and P/G Wire Co-optimization for Dynamic Noise," Proceedings of the 44th Design Automation Conference, pp. 162-167, 2007.

[17] http://www.gnu.org/software/glpk/

[18] http://www.gnu.org/software/octave/

[19] M. Zhao, R. Panda, S Sapatnekar and D. Blaauw, "Hierarchical Analysis of Power Distribution Networks," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 21, No. 2, pp. 159-168, Feb. 2002.

[20] J. Singh and S. S. Sapatnekar, "Partition-Based Algorithm for Power Grid Design Using Locality," IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, Vol. 25, No.4, pp. 664-677, April 2006.

[21] T. H. Hu, Integer Programming and Network Flows, Addison-Wesley Publishing, 1969.

[22] M. S. Tanaka, O. Takeuchi, H. Ogawa, H. Sasaki, H. Uchida and M. Ono, "Development of Chip-Package-Board Co-Design Techniques for Mixed-Signal Systems," IEICE Transactions, Vol. J93-C, No. 11, pp. 424-432, Nov. 2010.

[23] M. S. Tanaka, M. Kajita, N. Nakayama and S. Nakamoto, "A Method Using Circuits/Substrate Macro Modeling to Analyze Substrate Noise in a 3.2-GHz 350M-transistor Microprocessor," IEEE Custom Integrated circuits conference 2008, pp. 687-690, Sept. 2008.

[24] M. S. Tanaka, M. Kajita, N. Nakayama and S. Nakamoto, "Full Chip Circuits/Substrate Macro Modeling Method which Controls the Analyze Accuracy and CPU Time by Using Current Density," IEICE Trans. Fundamentals, Vol. E93-A, No.2, pp. 448-455, Feb. 2010.

[25] C. Ho, A. Ruehli and P. Brennan, "The Modified Nodal Approach to Network Analysis," IEEE Transactions on Circuits and Systems, Vol. 22, No. 6, pp. 504-509, June 1975.

[26] J. O. Attia, Electronics and Circuit Analysis Using Matlab, CRC press

[27] R. Sedgewick, Algorithms, Addison Wesley publishing company, 1988.

[28] M. Zhao, R. Panda, S. Sundareswaran, S. Yan and Y. Fu, "A Fast On-chip Decoupling Capacitance Budgeting Algorithm Using Macro Modeling and Linear Programming," Proceedings of the 43rd Design Automation Conference, pp. 217-222, 2006.

[29] J. Fu, et al. "A Fast Decoupling Capacitance Budgeting Algorithm for Robust On-chip Power Delivery," Proceedings of the ASP-DAC 43rd Design Automation Conference, pp. 505-510, 2004.

[30] H. Li, et al. "Partitioning-based Approach to Fast On-chip Decoupling Capacitor," Proceedings of the rd Design Automation Conference, pp. 170-175, 2005.

[31] Z. Qi, et al. "On-chip Decoupling Capacitor Budgeting by Sequence of Linear Programming," Proceedings of the 6th International Conference on ASIC, 2005, pp. 98-101, 2005.

[32] H. Su, et al. "Optimal Decoupling Capacitor Sizing and Placement for Standard Cell Layout Design," IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, Vol. 22, No.4, pp. 428-436, April 2003.

[33] X. Wang, et al. "An Efficient Decoupling Capacitance Optimization Using Piecewise Polynomial Models," Proceedings of the Design, Automation and Test in Europe, pp. 1190-1195, 2009.

[34] M. S. Tanaka, et al. "Greedy Optimization Algorithm for the Power/ground Network Design to Satisfy the Voltage Drop Constraint," IEICE Trans. Fundamentals, Vol. E94-A, No.4, pp. 1082-1090, Apr. 2011.

[35] S. Bronckers, G. Vandersteen, G. V. der Plas and Y. Rolain, "On the P+ Guard Ring Sizing Strategy to Shield Against Substrate Noise," on Proc. IEEE Radio Frequenxy Integrated Circuit, pp. 753-756, 2007.

[36] R. M. Vinella, G. V. der Plas, C. Soens, M. Rizzi and B. Castagnolo, "Substrate Noise Isolation Experiments in a 0.18um 1P6M Triple-well CMOS Process on a Lightly Doped Substrate," on Proc. Instrumentation and measurement techmology, pp. 1-6, May 2007.

[37] S. Bronckers, G. V. der Plas, G Vandersteen and Y ves Rolain, "Substrate Noise Coupling Mechanisms in Lightly Doped CMOS Transistors," IEEE Trans. on instrumentation and measurement, vol. 59, no.6, pp. 1727-1733, June 2010.

研究業績

雑誌論文

○ Mikiko Sode Tanaka, Nozomu Togawa, Masao Yanagisawa and Satoshi Goto, “Greedy Algorithm for the On-chip Decoupling Capacitance Optimization to Satisfy the Voltage Drop Constraint,” IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E94-A, No. 12, Dec. 2011.

○ Mikiko Sode Tanaka, Nozomu Togawa, Masao Yanagisawa and Satoshi Goto, “Greedy Optimization Algorithm for the Power/Ground Network Design to Satisfy the Voltage Drop Constraint,” IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E94-A, No. 4, pp. 1082-1090, Apr. 2011.

袖美樹子, 竹内修, 小川隼人, 佐々木英樹, 内田浩享, 小野光博, “アナデジ混載システムにおけるチップ、パッケージ、ボードの微小ノイズ設計 / 検証技術開発,”電子情報通信学会論文誌 C, Vol. J93-C, No. 11, pp. 424-432, 2010 年 11 月.

Mikiko Sode Tanaka, Mikihiro Kajita, Naoya Nakayama and Satoshi Nakamoto, "Full Chip Circuit/Substrate Macro Modeling Method Which Controls the Analysis Accuracy and CPU Time by Using Current Density," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E93-A, No. 2, pp. 448-455, Feb. 2010.

国際会議
（査読付き）

Mikiko Sode Tanaka, Masairo Toyama, Ryo Mori, Hidenari Nakashima, Masahiro Haida and Izumi Ooshima, “Early Stage Chip/package/Board Co-design Techniques for System-on-Chip,” Electrical Performance of Electronic Packaging and Systems (EPEPS), pp. 21-24, Oct. 2011.

○ Mikiko Sode Tanaka, Nozomu Togawa, Masao Yanagisawa and Satoshi Goto, “Substrate Noise Reduction Algorithm by Using the Current Flow Path,” International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), pp. 375-376, Jun. 2011.

Mikiko Sode Tanaka, O. Takeuchi, H. Ogawa, M. Ono, H. Uchida and H. Sasaki, “Noise Verification techniques for the mixed-signal chip/package/board of digital TV systems,” Electrical Performance of Electronic Packaging and Systems (EPEPS), pp. 205-208, Oct, 2010.

○ Mikiko Sode Tanaka, “Noise Analysis Method in Mixed-Signal Soc Floorplan Design for a Fast Success,” 15th Asia and South Pacific Design Automation Conference (ASP-DAC), Poster session, Jan. 2010.

Mikiko Sode Tanaka, Mikihiro Kajita, Naoya Nakayama and Satoshi Nakamoto, “A Method Using Circuit/Substrate Modeling to Analyze Substrate Noise in a 3.2GHz 350M Transistor Microprocessor,” Custom Integrated Circuits Conference(CICC), pp. 319-327, Sep. 2008.

Mikiko Sode Tanaka, Y. Miyazawa, H. Aizawa and M. Minowa, "Minimum Delay Placement with Influence of Nets and Hierarchical Clustering," Custom Integrated Circuits Conference (CICC), pp. 663-666, Sep. 1994.

Mikiko Sode Tanaka and Masaki Ishikawa, "A Multilayer Channel Router Based on Optimal Multilayer Net Assignment," International Conference on Very Large Scale (VLSI), pp. 301-310, Sep. 1993.

Mikiko Sode Tanaka, "A Multi-layer Channel Routing Algorithm," Joint Technical Conference on Circuits/Systems, Computers and Communications (JTC-CSCC), pp. 465-470, Jan. 1991.

国内会議

（査読付き）

袖美樹子, 古林紀哉, “多層チャネル配線のための割り当てネット選択アルゴリズム,” 電子情報通信学会回路とシステム軽井沢ワークショップ, pp. 85-90, 1993 年 4 月.

石塚昭, 岡本匠, 多和田茂芳, 袖美樹子, 高見沢一彦, “ゲートアレイ / セルベース IC 自動レイアウトシステム ～ GALET ～,” 情報処理学会 DA シンポジウム, pp. 137-142, 1994 年 8 月.

袖美樹子, 枝広正人, 吉村猛, “ゲート敷き詰め型ゲートアレイ用配置アルゴリズム,” 電子情報通信学会回路とシステム軽井沢ワークショップ, pp. 367-372, 1991 年 4 月.

袖美樹子, 枝広正人, “ゲートアレイレイアウトのためのチャネルへのネット割り当て手法によるグローバル配線,” 電子情報通信学会回路とシステム軽井沢ワークショップ, 1990 年 5 月.

講演

多和田茂芳, 水牧俊博, 田形充, 袖美樹子, 岩元圭一郎, 水沼貞幸, “GA/CBIC 自動レイアウトシステム GALET におけるタイミングドリブン自動配置手法,” 情報処理学会研究報告, 設計自動化研究会報告　94(93),　pp. 25-30, 1994 年 10 月.

多和田茂芳, 水牧俊博, 田形充, 袖美樹子, 岩元圭一郎, 水沼貞幸, “GA/CBIC 自動レイアウトシステム GALET におけるタイミングドリブン自動配置手法,” 電子情報通信学会技術研究報告, FTS, フォルトトレランスシステム　94(313), pp. 25-30, 1994 年 10 月.

多和田茂芳, 水牧 俊博, 田形 充, 袖 美樹子, 水沼 貞幸、石森 彰雄, “ゲートアレイ /CBIC 自動レイアウトシステム GALET(3)：自動配置手法,” 情報処理学会全国大会第 49 回平成 6 年後期 (6), pp. 119-120, 1994 年 9 月.

宮澤義幸, 袖美樹子, 石川正樹, “遅延考慮配置における遅延コストの動的反映方,” 電子情報通信学会 秋季大会, 1993 年 9 月.

袖美樹子, 宮沢義幸, “遅延考慮配置アルゴリズム,” 情報処理学会研究報告, 設計自動化研究会報告、 93(55), pp. 55-61, 1993 年 6 月.

袖美樹子, 吉村猛, “多層チャネルルータ,” 情報処理学会全国大会第 45 回　平成 4 年後期 (6), pp. 71-72, 1992 年 9 月.

袖美樹子, 吉村猛、” 多層チャネルルータ,” 情報処理学会研究報告, 設計自動化研究報告, pp. 31-36, 1992 年 5 月.

袖美樹子, “ゲートアレイレイアウトのための概略配線に関する一考案,” 情報処理学会全国大会 第 43 回 平成 3 年後期, 1991 年 10 月.

袖美樹子, 枝廣正人, 吉村猛, “ゲート敷き詰め型ゲートアレイ用配置アルゴリズム,”情報処理学会全国大会 第 42 回　平成 3 年前期 (6), pp. 236-237, 1991 年 2 月.

袖美樹子, 枝広正人, “遅延を考慮したチャネル配線手法,” 電子情報通信学会 秋季大会, 1990 年 10 月.

袖美樹子, 枝廣正人, 吉村猛、”階層クラスタリング手法を用いたゲートアレイ配置手法,” 情報処理学会全国大会 第 39 回 平成元年後期 (3), pp 1615-1616, 1989 年 10 月.

その他：**meeting**（査読付き）

Mikiko　Sode Tanaka, “Relative accuracy simulation method by using ADIT and CPU Distribution,” Mentor user meeting (Mentor U2U’08), Nov. 2008.

Mikiko　Sode Tanaka, “Reliability Simulation in Advanced LSI Design,” Mentor　user meeting (Mentor U2U’07), Mar. 2007.

：講演

袖美樹子, 佐々木英樹, “アナデジ混載システムにおけるノイズ低減の考え方,” 日本能率協会主催　EMC・ノイズ対策シンポジウム, 2011 年 7 月.

袖美樹子, “Eldo 活用事例紹介,” EDA Tech Forum, 2006 年 9 月.

袖美樹子, “ADvanceMS 適用事例,” EDA Tech Forum, 2005 年 9 月.

袖美樹子, “高周波向け設計環境 サブストレート関連を中心,” Cadence User　Group Meeting Japan (CUMJ), 2004 年 10 月.

袖　美樹子, ”Ｓｐｅｃｔｒｅを用いたＰＬＬ設計例,” イノテックケイデンスユーザーミーティン, 10 月, 2000 年

：雑誌

Mikiko Sode Tanaka, “Relative Accuracy Simulation Method by Using ADIT and CPU Distribution,” EE Times Mentor Graphics Technical Library, Nov. 2008.

袖美樹子, 横川慎二, 加藤一郎, 村井正宣, 関口亨, “先端 LSI 設計における信頼性シミュレーション,” Mentor　News & Views, 2007 年 10 月.

Mikiko Sode Tanaka, Masanobu Murai, Shinji Yokogawa, Toru Sekiguchi and Ichiro Kato, “Reliabililty-Simulation Environment Tackles LSI Design,” Chip Design Magazine, Jun. 2007.

袖美樹子, 横川慎二、加藤一郎, 村井正宣, 関口亨, “10 年後の動作をシミュレーションで確認する NEC エレクトロニクスが信頼性確保で新手法,” 日経マイクロデバイス, 2007 年 2 月.

静塚昇, 尾崎英晴, 高瀬宣言之, 袖美樹子, “大規模 ASIC におけるテスト技術,” NEC 技報 50(3), pp. 253-257, 1997 年 3 月.

：特許

Mikiko Sode and Yoichi Iizuka, “Flip Flop Circuit for Scan Test with Two Latch Circuits,” US Patent #6,006,348.