社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2003－ＳＬＤＭ－112 (33)
2003／11／28

クロストークノイズとシールド効果を考慮したクロスポイント割り当ての高速解法

木田 圭治[†,††] 朱 小科[†] 庄 昌文[†] 高島 康裕[††] 中武 繁寿[††]

† エスアイアイ・イーディーエー・テクノロジ株式会社 システム開発部
〒808-0135 福岡県北九州市若松区ひびきの2-5
†† 北九州市立大学 国際環境工学部
〒808-0135 福岡県北九州市若松区ひびきの1-1
E-mail: †{keiji.kida,xiaoke,changwen}@sii.co.jp, ††{takasima,nakatake}@env.kitakyu-u.ac.jp

あらまし 本稿では，ディープサブミクロンにおけるクロストークノイズとシールド効果を考慮したクロスポイント割り当て問題の高速解法を提案する．まず，概略配線セル境界を通過するネットの位置決め (クロスポイント割り当て問題) に対して，距離制約とシールド効果を考慮する $\{d,1\}$-ピッチ制約を導入する．この制約下でのクロスポイント割り当て問題は，許容解が存在する場合には，整数線形計画法に基づく手法により，厳密解を得ることができる．しかし，整数線形計画法による解法は多くの処理時間を必要とするため大規模な配線システムへの適用は困難である．本稿ではこの問題を解くための高速な発見的手法を提案する．実験において整数線形計画法に基づく手法と発見的手法を実データに適用し，発見的手法が充分高速に準最適解を導くことを確認した．

キーワード クロスポイント割り当て，クロストーク，シールド効果

A Fast Algorithm for Crosspoint Assignment under Crosstalk Constraints with Shielding Effect

Keiji KIDA[†,††], Xiaoke ZHU[†], Changwen ZHUANG[†], Yashuhiro TAKASHIMA[††], and Shigetoshi NAKATAKE[††]

† Systems Development Department, SII EDA Technologies Inc.
2-5, Hibikino, Wakamatsu-ku, Kitakyushu, Fukuoka 808-0135, Japan
†† Faculty of Environmental Engineering, The University of Kitakyushu
1-1, Hibikino, Wakamatsu-ku, Kitakyushu, Fukuoka 808-0135, Japan
E-mail: †{keiji.kida,xiaoke,changwen}@sii.co.jp, ††{takasima,nakatake}@env.kitakyu-u.ac.jp

Abstract In this paper, we present a fast algorithm for Crosspoint Assignment that takes into consideration crosstalk noise and shielding effects in deep sub-micron design. In our formulation, for crosspoint assignment problem(deciding a position where a global routing tree crosses a global routing cell boundary), we introduce $\{d,1\}$-pitch constraint that takes into consideration crosstalk with shielding effects. Under the constraints, an integer linear programming based algorithm can output an exact optimum solution if there exists a feasible solution. But, the integer linear programming based algorithm can not be applied to the typical routing system for the large scale design because it takes much calculating time. Therefore, we provide a fast heuristic algorithm for this problem. In experiments, we tested integer linear programming based algorithm and heuristic algorithm for industrial examples, and demonstrated that our heuristics ran quickly and attained near optimum solutions.

Key words crosspoint assign, crosstalk, shielding effect

1. はじめに

ディープサブミクロン時代の LSI 設計において, 配線のカップリングは, チップ動作を規定する上で支配的な要素となってきている. これはプロセスの進歩により, 配線が細くかつ高くなり, その結果, 配線容量におけるカップリング容量の比率が増大していることに起因する. 特に, 長距離を並走する配線のカップリングは, 信号遅延の増大を導くだけではなく, いわゆるクロストークノイズとして隣接信号の振る舞いにも影響を与え, チップを誤動作させることもある. そこで, 配線システムには, 正確にノイズをモデリングすると同時にクロストークノイズを確実に減らすための手法の確立が求められている [1], [2], [4]〜[8].

回路の大規模化に対し, 概略配線セル (GRC) と呼ばれる小領域を配線領域の基本単位とする階層的な配線システム用いられる. このシステムは, 以下の各段階から構成される. (i) チップ平面を格子状に配置された GRC の集合に分割する. (ii) GRC をノード, GRC の隣接関係を枝とする概略配線グラフを構成し, その上で, 各ネットの概略配線木を生成する [7]. (iii) GRC の境界を通過するネットの仮想端子の位置を決定する. これは, すなわち, GRC の境界と配線トラックの交点 (クロスポイント) に対し, ネットを割り当てる手続きである. この手続きをクロスポイント割り当て (CPA) と呼ぶ [1]〜[3]. 図 1 に CPA のイメージを示す. (iv) ネットを GRC 毎に分割し, 各 GRC 内で, 部分ネットの結線を行う [4], [5].

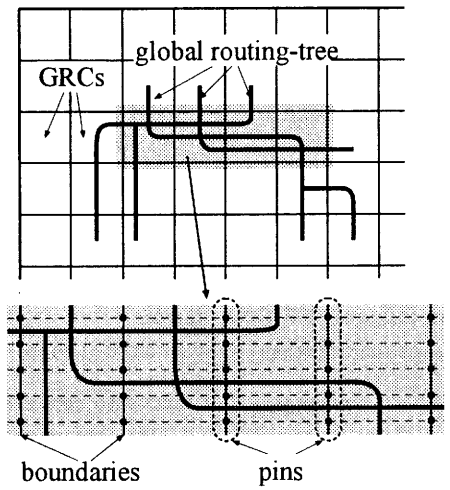

図 1　クロスポイント割り当ての例

本稿では, CPA においてクロストークノイズを回避する問題を扱う. 最近の研究 [6], [8] では, クロストークによる信号遅延やピークノイズを厳密に表現するためのモデリング方法が提案されている. 一方では, これらのモデルによる解析結果を利用したレイアウトアルゴリズムの開発も重要となってきている [8], [9].

ここでは, カップリング容量は, ネット対間の距離に反比例すると仮定する. よって, カップリング容量を抑えるために, 各

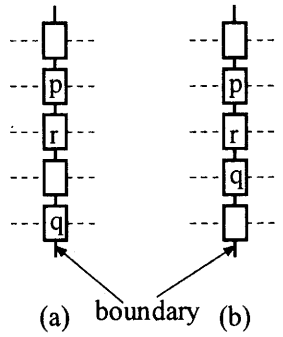

図 2　xs-制約を満たしたネット p,q の割り当ての例

ネット対に対し, それぞれを指定された距離 (ピッチ数) 以上離すことが要求される.

例えば, ある GRC 境界と交差し, 並走する 3 つのネット p, q, r を考える. (p,q), (q,r), (r,p) に対する距離制約の距離をそれぞれ 3-ピッチ, 1-ピッチ, 1-ピッチとする. この制約を満足する割り当ては図 2(a) のようになる. しかし, 実際には, r が p と q の間に置かれた場合, r によるシールド効果により p と q のカップリング容量は減少する. 従って, 図 2(b) のような割り当てにおいてもクロストークノイズは回避できる. このようにシールド効果を考慮に入れることにより, 高密度なレイアウトを得ることができる.

論文 [9] では, GRC の単一境界上での CPA 問題 (SBCPA 問題) に対し, クロストークノイズとシールド効果を考慮した制約 (xs-制約) の下での各ネット対に対する距離条件を与える $\{d,1\}$-ピッチ制約を導入している. ここで, $\{d,1\}$-ピッチ制約とは, 例えば, p と q の距離制約は, それらの間に他のネットの割り当てがなければ 3-ピッチ制約, 他のネットの割り当てがあれば, シールド効果により p,q のクロストーク制約が無視できることより, 1-ピッチ制約となることから, $\{3,1\}$-ピッチ制約となる. 本稿でもこの xs-制約下での SBCPA 問題を対象とする.

また, 論文 [9] では, xs-制約下での SBCPA 問題は NP-困難であることを示している. そのため解法として, 整数線形計画法に基づく手法を提案している. この手法は多項式時間で収束する保証はないが, 許容解があれば厳密な最適解を出力する. しかし, 多くの処理時間を必要とするため大規模な配線システムへの適用は困難である.

そこで, 本稿では, 実用上の観点から, 解の品質と計算時間のトレードオフをバランスさせた発見的手法の提案を行う. 提案手法は, 割り当てコストを最小化するネットを逐次選択し, 割り当てていくグリーディー戦略に基づくが, 各割り当て毎にシールド効果を観察し, ネット対の距離制約を更新していく工夫をしている. また, 計算複雑度は, 仮想端子数 n に対し $O(n^2 \log n)$ である.

我々は, 論文 [9] の整数線形計画法による解法 (*ILP*) と提案手法 (*Greedy*) を計算機実装し, 実データに適用する実験を行った. 評価は, 距離制約違反, 計算時間および割り当てコストとして行った. その結果, *Greedy* は *ILP* と同等の解を非常に高速に

出力した．これにより，提案手法の有効性を確かめた．

以降，2節では，SBCPA問題の定義を記述し，3節でこれに対する発見的手法を提案する．そして，4節で，提案手法の有効性を確かめるための実験とその結果を報告し，5節でまとめる．

2. SBCPA

単一GRC境界に対するCPA問題を *Single Boundary CPA* (SBCPA) 問題と呼ぶ．概略配線は既に終了しているものとする．つまり，各GRCの境界に対し，どのネットが通過しているかは決定されている．

ネット対のカップリング容量はその対間の距離に換算でき，容量は距離に反比例すると考えることができる．つまり，GRC境界におけるクロストーク制約とは，ネット対をそれぞれ指定された距離 (ピッチ数) 以上離すということである．

SBCPAの入力は以下の通りである．

（1） 境界と交差するネット集合 $\mathcal{N} = \{n_i | 1 \leqq i \leqq \nu\}$,

（2） 境界上の仮想端子集合 $\mathcal{T} = \{j | 1 \leqq j \leqq \tau\}$,

（3） $1 \leqq i \leqq \nu$ and $1 \leqq j \leqq \tau$ のときにネットと仮想端子の対に対して，割り当てコスト $c(i,j)$.

（4） $1 \leqq i, i' \leqq \nu$ のときにネット対に対する距離制約 $d(i,i')$. ここで，もし，n_i と $n_{i'}$ の間にネットが存在しないときは，それらのネットの距離は $d(i,i')$ 以上必要であるが，存在するときは距離は1以上で十分である．この制約を $\{d,1\}$-ピッチ制約と呼ぶ．

出力は $\{d,1\}$-ピッチ制約のもとで全体の割り当てコストを最小にするようなネットの仮想端子への割り当て $a : \mathcal{N} \to \mathcal{T}$ である．ここでの割り当てコストは，配線長を最小化するように定義されている．上記の定式化は距離制約および $\{d,1\}$-ピッチ制約を除き 論文[1] のものと同じである．なお，付録に論文[9]で紹介されているSBCPAの整数線形計画法での解法を示す．

3. 高速グリーディーアルゴリズム

[9] で提案された整数線形計画法での解法は厳密な最適解を得ることができるが，多くの処理時間を要するため，実際の大規模な配線システムには適用できないと考えられる．そこで，実用上の観点から解の品質と計算時間のトレードオフをバランスさせたアルゴリズム *Greedy* を提案する．*Greedy* の概要を以下に示す;

（1） 未割り当てのネットに対し，未割り当て仮想端子に対する割り当てコストの最大値と最小値の差を計算する．

（2） 最大のコスト差を持つネットを選択し，xs-制約を満たした中で，コスト最小の未割り当て仮想端子に割り当てる．

（3） 上記 (1), (2) を割り当てられるネットが無くなるまで繰りかえす．

このアルゴリズムは，目的関数への影響の大きいネットはより早く割り当てるという教訓的な戦略によるものである．この戦略は，その有効性から，幾つかのグリーディーアルゴリズムにおいてもしばしば登場する．

Greedy の詳細は以下の通りである．

アルゴリズム: Greedy

（1） 三つの配列を用意する;

$\mathcal{Q}$: ν 行と τ 列からなる実数値の二次元配列．i 行 j 列 の要素は $q_{i,j}$ で記述される．$q_{i,j}$ は $c(i,j)$ で初期化される．

$\mathcal{Y}$: サイズが τ の二値の一次元配列．要素は y_j で記述される．仮想端子 j がいずれかのネットに割り当てられた場合，$y_j = 1$ となる．それ以外の場合は $y_j = 0$ である．各要素は 0 で初期化される．

$\mathcal{Z}$: サイズが ν の実数値の一次元配列．要素は z_i で記述される．初期値は $z_i = \max_j q_{i,j} - \min_j q_{i,j}$ とする．

（2） $q_{i,j} = +\infty, -\infty$ or $\pm\infty \mid \forall j$ ならば $z_i = -1$ とする．これは，制約を満たす仮想端子が存在しないことを意味する．また，$q_{i,j} \neq +\infty, -\infty$ nor $\pm\infty$, となる仮想端子が一つだけ存在する場合には，$z_i = \infty$ とする．これは，(現状では) 制約を満たす仮想端子は一つだけであることを意味する．それ以外の場合は，$z_i = \max_j q_{i,j} - \min_j q_{i,j}$ とする，ただし，$q_{i,j}$ は $+\infty$, $-\infty$, または，$\pm\infty$ である．

ここで，$\mathcal{Q}$ と $\mathcal{Z}$ において，既に割り当てられているネットの要素はステップ (2)-(4) において無視される．

（3） $\mathcal{Z}$ 中の最大値に相当するネット n_i を選択する．

ネット n_i を $q_{i,j} = \min_{\forall k} q_{i,k}$ である仮想端子 j に割り当て，y_j に 1 をセットする．

n_i を $\mathcal{N}$ から取り除き，$\mathcal{N} = \phi$ なら終了する．

（4） $\mathcal{Q}$ を以下のように更新する．

$\mathcal{N}$ の各ネット $n_{i'}$ に対して，f が 0 になるまで以下の (a) と (b) を繰り返す．初期値として，$\delta = d(i,i')$, $j' = j$, および $f = 1$ をセットする．

（a） j' を 1 増加する．

（b） $y_{j'} = 1$ ならば $f = 0$ とする．それ以外の場合は，次の各値をセットする．

$$f = \begin{cases} 0 & \text{if } q_{i',j'} = +\infty \text{ and } |j' - j| \geqq \delta \\ 0 & \text{if } q_{i',j'} = c_{i',j'} \text{ and } |j' - j| \geqq \delta \\ 1 & \text{otherwise} \end{cases}$$

$$q_{i',j'} = \begin{cases} c_{i',j'} & \text{if } q_{i',j'} = -\infty \text{ and } |j' - j| \geqq \delta \\ \pm\infty & \text{if } q_{i',j'} = +\infty \text{ and } |j' - j| < \delta \\ +\infty & \text{if } q_{i',j'} = \pm\infty \text{ and } |j' - j| \geqq \delta \\ -\infty & \text{if } q_{i',j'} = c_{i',j'} \text{ and } |j' - j| < \delta \end{cases}$$

同様に，j' を減少させながら，$q_{i,j}$ を更新する．(そこでは，$+\infty$ と $-\infty$ はお互いに置き換わる)．

ステップ (2) に戻る． □

Greedy において，$\mathcal{Z}$ は各ネットのコストの差の保持に用いられ，$\mathcal{Q}$ の要素 $q_{i,j}$ は距離制約違反回避のために用いられ，$c(i,j)$, $-\infty$, $+\infty$, $\pm\infty$ のうちのいずれかである．ここで，$-\infty$ ($+\infty$, $\pm\infty$) は左 (右, 両) 側の仮想端子に割り当てられたネットにより，距離制約違反が発生することを意味する．

次に, アルゴリズムの計算複雑度を考察する. ステップ (1) は配列 $\mathcal{Q}$ の初期化により, $O(\nu \cdot \tau)$ である. ステップ (2) では $\max_{\forall k} q_{i,k}$ および $\min_{\forall k} q_{i,k}$ を得る必要がある. この計算複雑度は, 各ネットに対してヒープを用いることにより $O(\nu \cdot log\tau)$ である. ステップ (3) は $\mathcal{Z}$ から最大値を選択するので, $O(\nu)$. ステップ (4) は二重ループで構成されている. 外側のループはネットに対するもので, 内側は $\{d,1\}$-ピッチ制約に対するものである. よって, この部分の計算複雑度は, $O(\nu \cdot d)$ となる. なお, d は定数とみなすことができる. ステップ (2), (3), および (4) は ν 回繰り返され, また, $O(\nu) = O(\tau)$ であるため, 全体の計算複雑度は $O(\tau^2 \log \tau)$ となる.

理解を深めるために, *Greedy* の動作の例として, 最初の二回の繰り返しとその結果についてを説明する.

例

4 ネット $\{n_0, n_1, n_2, n_3\}$ と 5 仮想端子 $\{0,1,2,3,4\}$ が与えられている. また, 割り当てコストと距離制約が以下のように与えられている.

$c(i,j) =$

$n\backslash p$	0	1	2	3	4
0	2	−3	−1	1	1
1	0	1	−2	2	4
2	1	0	2	−1	0
3	−2	1	2	3	1

$d_{i,j} =$

$n\backslash n$	0	1	2	3
0	0	3	1	1
1	3	0	1	2
2	1	1	0	1
3	1	2	1	0

$\mathcal{Q}, \mathcal{Y}, \mathcal{Z}$ の初期値は以下の通りである.

$\mathcal{Q}_0 =$

$n\backslash p$	0	1	2	3	4
0	2	−3	−1	1	1
1	0	1	−2	2	4
2	1	0	2	−1	0
3	−2	1	2	3	1

$\mathcal{Y}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix}$, $\mathcal{Z}_0 = \begin{bmatrix} 5 & \underline{6} & 3 & 5 \end{bmatrix}$.

はじめに, $\mathcal{Z}_0$ の中で最大は z_1 であるので, 対応するネット n_1 を選択する. そして, $\mathcal{Q}_0$ の第 1 行では $q_{1,2}$ が最小よりネット n_1 を仮想端子 '2' に割り当てる. $\mathcal{Q}_0$, $\mathcal{Y}_0$ および $\mathcal{Z}_0$ は, それぞれ $\mathcal{Q}_1$, $\mathcal{Y}_1$ および $\mathcal{Z}_1$ に更新される.

$\mathcal{Q}_1 =$

$n\backslash p$	0	1	2	3	4
0	$+\infty$	$+\infty$	*	$-\infty$	$-\infty$
1	*	*	*	*	*
2	1	0	*	−1	0
3	−2	$+\infty$	*	$-\infty$	1

$\mathcal{Y}_1 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix}$, $\mathcal{Z}_1 = \begin{bmatrix} -1 & * & 2 & \underline{3} \end{bmatrix}$

なお, '*' は各テーブルにおいて無視される.

つぎに, ネット n_3 を選択し仮想端子 '0' に割り当てる. $\mathcal{Q}_1$, $\mathcal{Y}_1$ および $\mathcal{Z}_1$ は以下のように更新される.

$\mathcal{Q}_2 =$

$n\backslash p$	0	1	2	3	4
0	*	$+\infty$	*	$-\infty$	$-\infty$
1	*	*	*	*	*
2	*	0	*	−1	0
3	*	*	*	*	*

$\mathcal{Y}_2 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \end{bmatrix}$, $\mathcal{Z}_2 = \begin{bmatrix} -1 & * & 1 & * \end{bmatrix}$.

最終的な割り当て結果は, $a(0) = 4$, $a(1) = 2$, $a(2) = 3$ および $a(3) = 0$ となる. ここで, *Greedy* がシールド効果を考慮していることに着目する. 例えば, n_0 と n_1 に対して $d(0,1) = 3$ であるが, n_2 がシールドとして存在するため, $a(0) = 4$ と $a(1) = 2$ が許容される.

4. 実　　験

アルゴリズム の性能評価を行うために *Greedy* を Sun Blade 2000 (900MHz) 上に C++ で実装した. また, *Greedy* の解の品質と計算時間の比較を行うために論文 [9] の整数線形計画法による解法 (*ILP*) の実装も行った. なお, *ILP* の詳細は, 付録として記述した. また, *ILP* の実装には, ILP パッケージとして lp_solve_4.0 を使用した.

実データとして 16,000 ネット, 40,000 インスタンスからなる LSI において, 混雑度の高い GRC 境界を抽出して用いた. 各ネットペアに対する $\{d,1\}$-ピッチ制約は, 既存の遅延抽出ツールからの情報に基づいて手作業で付加した. なお, クリティカルパスに対しては, より高い距離制約を付加した. その一例として 26 ネットの単一 GRC 境界のデータでは, $\{5,1\}$-, $\{4,1\}$-, $\{3,1\}$-, および $\{2,1\}$-ピッチ制約数が, それぞれ 2, 3, 11, および 33 となった.

はじめに, *Greedy* と *ILP* に対して, 仮想端子数が 20 から 40 の 11 個のデータに対して, それぞれ xs-制約を考慮した *Greedy* と考慮しない *Greedy*, xs-制約を考慮しない *ILP* について, 実験を行なった. (xs-制約を考慮した *ILP* は仮想端子数が 20 以上のデータにおいては, 実時間で収束しなかった.)

結果を 表 1 に示す. ここで, (a),(b) および (c) は, それぞれ xs-制約を考慮した *Greedy*, 考慮しない *Greedy*, xs-制約を考慮しない *ILP* を示す. xs-制約を考慮しない *Greedy* のコストは xs-制約を考慮しない *ILP* とほぼ同等である. すなわち, *Greedy* の最適化能力は, ほぼ *ILP* と同等であると考えられる. また, xs-制約を考慮した *Greedy* では殆どの例で距離制約違反が 0 であるにも関わらず, 計算時間において, xs-制約を考慮しない *Greedy* と同等である. これは, アルゴリズムでの距離制約違反回避が非常に効率的であることを示している.

次に, *Greedy* と *ILP* に対して, 仮想端子数が 10 の小規模な実験データでテストした. 結果を 表 2 に示す. 両方のアルゴリズムにおいて xs-制約を考慮している. ここで, *ILP* は常に最適解を出力する. 一方, *Greedy* は一例を除いて xs-制約を満足する解を *ILP* と比較して非常に高速に出力する. また, コストに関しても, ほとんどの場合, *ILP* と同様の解を出力している.

これらの実験結果より, *Greedy* は xs-制約が付加された場合でも, 許容できる準最適解を非常に高速に出力可能であると結論付けることができる.

5. おわりに

階層的な戦略を用いる配線システムに適用するための, クロストーク制約とシールド効果を同時に考慮できる単一 GRC 境界上での CPA の高速解法 *Greedy* を提案した. この手法は, 仮

表 1　解の品質と計算時間の比較結果：('w/ const.') は xs-制約を考慮した場合, ('w/o const.') は xs-制約を考慮しない場合.

no.	#pins	#nets	*Greedy*					*ILP*	
			#violations	cost		time(msec.)		cost	time(msec.)
			w/ const.	(a) w/ const.	(b) w/o const.	(a) w/ const.	(b) w/o const.	(c) w/o const.	(c) w/o const.
1	20	13	0	513	103	0.091	0.09	81	50
2	20	10	1	-568	-568	0.059	0.059	-568	20
3	20	12	0	-6,124	-6,124	0.077	0.078	-6,124	30
4	30	18	0	-9,812	-10,374	0.228	0.227	-10,374	70
5	30	17	0	-2,629	-1,945	0.204	0.203	-2,677	90
6	30	20	0	-11,384	-14,231	0.257	0.262	-14,231	70
7	30	20	1	-13,387	-13,991	0.263	0.265	-14,260	120
8	40	26	0	-16,706	-14,575	0.562	0.562	-16,706	220
9	40	27	0	-16,437	-19,562	0.568	0.578	-19,562	180
10	40	26	0	-24,279	-24,279	0.537	0.538	-24,279	170
11	40	26	0	-17,629	-17,629	0.535	0.539	-17,629	170

表 2　xs-制約を考慮した場合の *Greedy* と *ILP* の比較結果

no.	#pins	#nets	#violations		cost		time (msec.)	
			Greedy	*ILP*	*Greedy*	*ILP*	*Greedy*	*ILP*
12	10	6	0	0	1,903	1,903	0.016	288,130
13	10	8	0	0	2,677	2,677	0.024	258,590
14	10	7	0	0	776	769	0.02	681,090
15	10	8	0	0	520	520	0.025	856,620
16	10	7	0	0	2,664	2,628	0.02	147,240
17	10	8	0	0	1,786	1,786	0.024	265,090
18	10	7	0	0	172	172	0.02	177,400
19	10	7	0	0	-1,144	-1,144	0.02	94,720
20	10	8	1	0	100	1,000	0.022	298,550
21	10	8	0	0	1,857	1,809	0.024	360,770

想端子数を n とした場合, 計算複雑度が $O(n^2 \log n)$ である.

実験において, 整数線形計画法による解法 (*ILP*) と提案手法 (*Greedy*) を実装し, 遅延情報から抽出した距離制約を用いて実データに適用した. この二つのアルゴリズムを距離制約違反, 解の品質 (コスト) および計算時間の観点から比較し評価を行った. 実験から *Greedy* が非常に高速であり, xs-制約が付加された場合でも, 許容できる準最適解を非常に高速に出力するという結果が得られた.

今後の課題として, *Greedy* を実際の配線システムにおいて, 複数境界上での CPA に拡張することが挙げられる. このためには, (i) ネット対に対し, xs-制約をどのように付加するのか, (ii) GRC 境界おいて, $\{d, 1\}$-ピッチ制約をどのように付加するのか, という問題を解決する必要がある.

謝　辞

本研究は、文部科学省の北九州地域と福岡地域の広域知的クラスター創成事業の支援による。

文　献

[1] W.-C. Kao and T.-M. Parng, Cross Point Assignment with Global Rerouting for General-Architecture Designs, *IEEE Trans. on CAD of ICs and Systems*, 14(3):337-348, 1995.

[2] C.-C. Chang and J. Cong, Pseudo Pin Assignment with Crosstalk Noise Control, *IEEE Trans. on CAD of ICs and Systems*, 20(5):598–611, 2001.

[3] H.-P. Tseng and L. Scheffer and C. Sechen, Timing and Crosstalk Driven Area Routing, *In Proc. of Design Automation Conf.*, 378–381, 1998.

[4] T. Gao and C. L. Liu, Minimum crosstalk switchbox routing, *Integration, The VLSI Journal*, 19(3):161–180, 1995.

[5] T. Gao and C. L. Liu, Minimum crosstalk channel routing, *IEEE Trans. on CAD of ICs and Systems*, 15(5):465–474, 1996.

[6] T. Xue and E. S. Kuh and D. Wang, Post global routing crosstalk risk estimation and reduction, *In Proc. of Intl. Conf. on CAD*, 302–309, 1996.

[7] H. Zhou and D. F. Wong, Global routing with crosstalk constraints, *In Proc. of Design Automation Conf.*, 374–377, 1998.

[8] J. Cong and D. Z. Pan and P. V. Srinivas, Improve Crosstalk Modeling for Noise Constrained Interconnect Optimization, *In Proc. of Asia South Pacific Design Automation Conf.*, 373–378, 2001.

[9] Yasuhiro Takashima and Shigetoshi Nakatake and Yoji Kajitani, An ILP-Based Algorithm for Crosspoint Assignment under Crosstalk Constraints with Shielding Effects, *The 16th Workshop on Circuits and Systems in Karuizawa.*, 213–218, 2003.

付　　録

SBCPA に対する整数線形計画法による解法を記述する.

まず, 各ネット n_i と各仮想端子 j に対して, n_i の j への割り当てをモデル化するため, 0-1 の整数変数 $x_{(i,j)}$ を導入する. ここで, $x_{(i,j)}=1$ は n_i が j に割り当てられること, $x_{(i,j)}=0$ は割り当てられていないことを意味する.

この計画法での目的は次の関数を最小化することである:

$$\sum_i \sum_j c(i,j)x_{(i,j)}.$$

整数線形計画法の制約は以下の線形式で定式化される.

1-仮想端子制約　各ネット n_i が丁度一つの仮想端子に割り当られることを保証する.

$1 \leqq i \leqq \nu$ である全ての i に対して,

$$\sum_j x_{(i,j)} = 1.$$

1-ネット制約　各仮想端子 j が高々一つのネットに割り当てられることを保証する.

$1 \leqq j \leqq \tau$ であるすべての j に対して,

$$\sum_i x_{(i,j)} \leqq 1.$$

$\{d,1\}$-ピッチ制約　ネット n_i と $n_{i'}$ の距離が $d(i,i')$ 以上であるか, n_i と $n_{i'}$ の間に他のネットが少なくとも一つ存在することを保証する. 次の 3 種の 0-1 の整数変数: $e(i,i',j,j')$, $s(i,i',j,j')$, および $v(i,i',j,j')$ を導入する.

$e(i,i',j,j')$ は n_i と $n_{i'}$ の割り当ての状況を表現する. n_i, $n_{i'}$ がそれぞれ 仮想端子 j, j' に割り当てられた場合は $e(i,i',j,j')=1$, そうでない場合は $e(i,i',j,j')=0$ である. すなわち,

$$0 \leqq \frac{x_{(i,j)}+x_{(i',j')}}{2} - e(i,i,j,j') < 1.$$

$s(i,i',j,j')$ はネット n_i, $n_{i'}$ が, 仮想端子 j, j' に割り当てられたときにシールド効果が働くかどうかを表現する. シールド効果が働くときは $s(i,i',j,j')=1$, そうでないときは, $s(i,i',j,j')=0$ である. すなわち,

$$0 < \frac{\sum_{j \leqq h \leqq j'} \sum_{n_{i''},i'' \neq i,i'} x_{(i'',h)}}{|j-j'|+1} + s(i,i',j,j') \leqq 1.$$

$v(i,i',j,j')$ については, $e(i,i',j,j')=1$ かつ $s(i,i',j,j')=1$ ならば $v(i,i',j,j')=1$, そうでなければ, $v(i,i',j,j')=0$ である. すなわち,

$$0 \leqq \frac{e(i,i',j,j')+s(i,i',j,j')}{2} - v(i,i',j,j') < 1.$$

以上より, n_i と $n_{i'}$ の間の $\{d,1\}$-ピッチ制約は次のように表現できる.

$$\sum_j \sum_{\max\{1,j-d(i,i')\} \leqq j' \leqq \min\{\tau,j+d(i,i')\}} v(i,i',j,j') = 0.$$

出力は $a(i) = \sum_j j x_{(i,j)}$ となる. 上記の式を ILP を用いて解くことにより SBCPA の最適解を得る.