設 計 自 動 化 87－5
（1998．1．30）

タイミング制約を伴う概略配線問題に対する バッファ挿入と配線幅の選択を許したスタイナ木構成手法

大釜浩介　小出哲士　若林真一
広島大学工学部
〒739-8527 東広島市鏡山一丁目4番1号
E-mail:{kosuke, koide, wakaba}@ecs.hiroshima-u.ac.jp

本稿ではVLSIレイアウト設計における配線木生成において，複数の部分木の配線トポロジを保持しながら配線木へのバッファ挿入，配線幅拡大を同時に考慮したスタイナ木生成手法を提案する．提案手法では，初期配線トポロジは仮定せず，ソースを原点とするグリッド平面上に与えられたシンクノードの座標位置，およびソースより各シンクまでの要求遅延時間を元に配線木の構築を行なう．また，配線木の構築と同時に木の枝に対してバッファ挿入，配線幅拡大を行なう．さらに提案手法では，シンクからソースに向けてボトムアップに配線木を構築していく際に，構築過程で生成される部分木は1形状に限定せず，複数の部分木の生成を許すことで形状の事なる複数の配線木を構築し，最終段階で最もスラックの大きい節点をソースからシンクまでバックトレースする事により，最良の配線木を決定する．全ての可能な部分木の組合せを計算すると膨大な計算時間がかかるため，提案手法では配線木の構築過程で効果的な枝刈り手法を用い，かつ提案手法を階層的に適用する事により実用的な時間で配線木を求める事が出来る．

A Rectilinear Steiner Tree Construction Algorithm with Simultaneous Buffer Insertion and Wire Sizing

Kosuke OGAMA, Tetsushi KOIDE and Shin'ichi WAKABAYASHI
Faculty of Engineering, Hiroshima University
4-1, Kagamiyama 1 chome, Higashi-Hiroshima 739-8527 JAPAN

This paper presents a rectilinear Steiner tree construction algorithm considering multiple topologies of subtrees with simultaneous buffer insertion and wire sizing during tree construction. The proposed algorithm does not require an initial tree topology, but only requires positions of source and sink nodes on the grid plane and required arrival times at each sink. While the algorithm constructs rectilinear Steiner trees, buffer insertion and wire sizing are also considered. Moreover, during the tree construction from sinks to the source with a *bottom_up* manner, the algorithm restricts the number of subtree topologies which are produced during the tree constructing process and produces different tree toplogies. After constructing all trees,we can find a maximum slack solution by backtracing the tree with the maximum slack value. Since computing all possible combinations wastes much run time this paper presents some efficient pruning methods and a hierachical approach for large scale data to reduce the computation time.

1 はじめに

VLSI レイアウト設計は近年の微細化技術の進歩に伴ってますます複雑になってきている．チップ内の配線に関しても，最近では配線幅 $0.25\mu m$ ルールによる配線も実用化され，将来に向けては，さらに配線幅は小さくなっていくものと予想されている．また，チップの動作も年々高速化してきている．このような近年の半導体技術の進歩にともない，レイアウト設計における配線設計においてもチップ面積最小化のみを目的として総配線長を最小化する従来の配線手法では，現在要求されている高速なチップを設計することが困難になりつつある．

チップ内の遅延において，従来は配線遅延よりもゲート遅延が重要な要因となっており，配線遅延は無視できる範囲にあった．しかし，前述のようにチップの微細化が進みそれにともなって配線幅も急激に減少してきた現在のチップにおいては，配線抵抗は配線幅に反比例して大きくなるため，ゲート遅延より配線遅延の方が回路全体の遅延の中で占める割合が増大し，配線遅延が最も重要な要素となってきた．最近では配線遅延が全体の遅延の 50 % を超えてきている．この様な状況から配線遅延を陽に考慮した効率的なレイアウト設計手法が不可欠となってきている [4]．

概略配線設計の研究においても，近年，タイミング制約を考慮した概略配線手法が盛んに研究されている．配線遅延の減少のため，概略配線の配線経路上にバッファを挿入したり，幅広配線を行なうことが一般的である [1〜4]．バッファ挿入については，従来手法の多くは配線トポロジが与えられているものとし，タイミングのクリティカルなパスに対してバッファを挿入することにより配線遅延の減少を計っている [3]．また，配線形状は与えられず，配線トポロジの決定とバッファ挿入，配線幅拡大を考慮している手法として [1] がある．文献 [1] では，与えられたネットに対して配線トポロジを決定するために，*Heuristic A-tree* という手法を用いている．この手法は，各シンクに対し，ソースから遠いシンクより順にペアを作っていき，ボトムアップに配線経路を決定していく．しかし，この手法では極端に長いパスが出来てしまう場合がある．また，文献 [2] では，シンクの順序の入れ換えを考慮して配線経路を構成していく手法を提案している．しかしながら，配線トポロジの決定とバッファ挿入や配線幅拡大を 2 段階に分けて行なうこれらの手法においては必ずしも最適な概略配線経路を生成できない場合がある．

本稿では，初期配線トポロジは仮定せず，配線木を構築しながら同時に木の枝に対して，バッファ挿入，配線幅拡大を行なう手法を提案する．提案手法では，シンクの最小スラックを最大にするという条件の下で総配線キャパシタンスの最小化を行なう．また，大規模データに対応するため，提案手法を階層的に適用する手法も同時に提案する．

以降では，2 節で問題を定式化，3 節で提案手法を説明，4 節で提案手法の拡張を行なう．5 節でシミュレーション実験，6 節で本研究をまとめる．

2 準備

2.1 遅延モデル

本稿では，配線の遅延モデルとしては Elmore 遅延モデル，バッファの遅延モデルとしては RC モデルを採用する [1]．

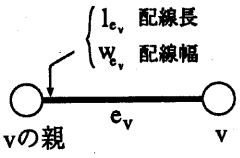

図 1: 枝の説明

配線 e_v について，節点 v の親から節点 v までの配線とする．$l_{e_v}, w_{e_v}, c_{e_v}, r_{e_v}$ をそれぞれ配線長，配線幅，キャパシタンス，及び抵抗とする (図 1)．c_a, c_f, r_0 を単位幅，及び単位長さ当たりのエリアキャパシタンス，フリンジキャパシタンス，及び抵抗とする．$T(v)$ を v を根とする部分木，$c(T(v))$ を $T(v)$ 中の *dc_connected* 部分木のキャパシタンスとする [1]．ここで *dc_connected* 部分木とは節点 v に直接接続された部分木を意味する．*dc_connect* 部分木のキャパシタンスは図 3 の $c(T(v))$ にあたる部分で，節点 v に注目すると節点 v から v の親までの枝，節点 v から左右の子 u, w までの枝のキャパシタンス の総和となる．d_b, r_b をバッファ b の固有の遅延，及び出力等価抵抗，c_l をバッファ b のロードとする．配線遅延を $D_{wire}(e_v)$，バッファ遅延を $D_{buff}(b, c_l)$ とすると，

$$c_{e_v} = (c_a \cdot w_{e_v} + c_f) \cdot l_{e_v} \tag{1}$$

$$r_{e_v} = r_0 \cdot l_{e_v} / w_{e_v} \tag{2}$$

$$D_{wire}(e_v) = r_{e_v} \cdot ((c_{e_v}/2) + c(T(v))) \tag{3}$$

$$D_{buff}(b, c_l) = d_b + r_b \cdot c_l \tag{4}$$

と表すことが出来る．

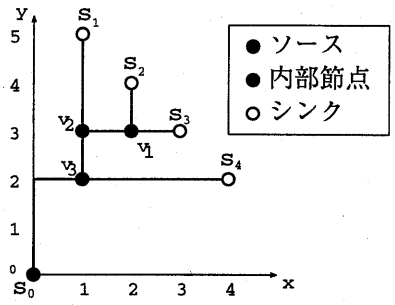

図 2: A-tree のアルゴリズム

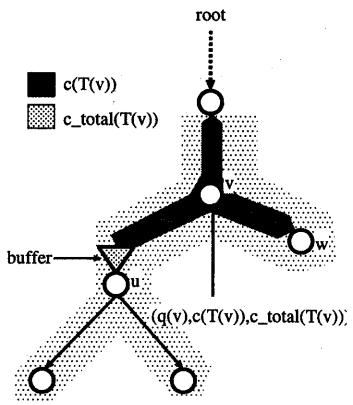

図 3: 部分木

2.2 遅延スラック

提案手法では，初期配線トポロジは与えられず入力としてはソース s_0, シンク $s_1, \cdots, s_n$の端子集合と各シンク s_iでの要求遅延時間 $RAT(s_i)(1 \leq i \leq n)$) とする．出力は配線幅拡大とバッファ挿入を行なったスタイナ木 $T(s_0)$ である．また本稿では簡単化のため，各シンクはソース s_0を原点とした時に第 1 象限に分布しているものとしている．シンク $s_i \in S(S$は端子集合) の要求遅延時間を $RAT(s_i)$ とし，ソース s_0の要求遅延時間を $q(s_0) = 0$ とする．配線木の節点 vは端子集合 Sと内部節点集合 Iの和集合で表され，$v \in S \cup I$である．節点 vからシンク s_iまでの遅延を $Delay(v, s_i)$ とする．vでの遅延スラックを $q(v)$ とすると，$q(v)$ は以下のように表すことが出来る．

$$q(v) = min_{s_i \in T(v)}(RAT(s_i) - Delay(v, s_i)) \quad (5)$$

この遅延スラックは節点 vの遅延時間の余裕を表し，$q(v) < 0$ となる節点はタイミング制約を違反している事を示す．

2.3 問題の定式化

本稿ではソースの遅延スラックを最大化した上で総配線キャパシタンス最小化を目的とした定式化を行なう．ソースから各シンクまでの遅延時間を制約とする．

【入力】: 端子集合 $S = \{s_0, s_1, \cdots, s_n\}$
(s_0, シンク $s_1, , ..., s_n$),
バッファライブラリ B, 配線幅集合 W,
シンク s_iの要求遅延時間 $RAT(s_i)$.
【出力】: 配線幅拡大とバッファ挿入後の
スタイナ木 $T(s_0)$
【目的関数】: $q(s_0)$ の最大化の上で
総キャパシタンスの最小化,
【制約条件】: シンク s_iの遅延制約
$(Delay(s_0, s_i) \leq RAT(s_i), \forall s_i \in S)$

3 提案アルゴリズム

3.1 A-tree アルゴリズム

本稿で提案する概略配線アルゴリズムは，配線木の構築に当たっては文献 [1] の手法を参考にしている．本節では，文献 [1] で述べられている *A-tree* について、簡単な例を用いて説明する．

図 2に示すように，シンク s_1, s_2, s_3, s_4, とソース s_0 の位置が与えられているとする．シンク s_i, s_jの x, y座標をそれぞれ $x(s_i), y(s_i)$ 及び $x(s_j), y(s_j)$ とすると *A-tree* では，

$$max\{min(x(s_i), x(s_j)) + min(y(s_i), y(s_j))\} \quad (6)$$

となるシンクペアを見つけ，それぞれのノードから x 軸および y軸に平行な直線をひき，ソースに近い方の交点を内部節点としてシンクペアをマージする．マージしたシンクペアは消去し，新たに内部節点と残りのシンクとで同様の操作を繰り返す．図 2ではまず s_2, s_3のペアがマージされた後，s_1, v_1のペアがマージされ，最後に v_2, s_4ペアがマージされる．このように，木を構成していくと一意に木は決定できるが，(v_3, s_4) の枝の様な極端に長い枝が出来る可能性がある．

そこで，本稿で提案するアルゴリズムでは，配線木を構築する際に，文献 [1] と同様にスラックが最大になるようにバッファ挿入，配線幅拡大を行なうが，1つ

の木を構築していくのではなく複数の木を構築していく．そのため，*A-tree* よりもトポロジの決定に自由度を与えることになり，さらによい解を得る事が期待できる．なお提案手法では，説明の簡単のために構成される木の構造は 2 分木とする．

3.2 背景

従来手法 [1] では，配線木のトポロジは配線木生成過程で一意に決定される．文献 [1] の手法はシンクからソースに向けてボトムアップに配線木を生成していくため，何らかの指標を用いて節点のペアを選択する際，節点ペアの候補の中で評価が同じものが存在した場合，必ずどれか一つのペアを選択しなければならない (図 4)．しかし，アルゴリズムの性質上配線木の構造を先読みする事は出来ないため，例えば図 4で v_1, v_2 のペアが選ばれたとしても，最終的に生成される木で評価すれば，v_2, v_3のペアを選択していた方が良い解が得られる場合もあり得る．生成途中の部分木の評価が同じでも最終的に作られる木の評価が同じであるとは限らない．そこで本稿では，どちらのペアもアルゴリズムが終了するまで保持する事により，この様な問題を回避する．また，全ての組合せを保持すると膨大な計算時間が必要となるため，効率的な枝刈り手法も提案する．

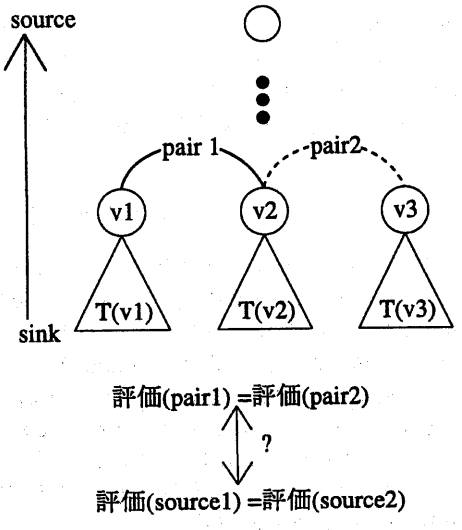

図 4: 手法 [1] との比較

3.3 複数の部分木の構成法

本節では，複数の部分木の構成法について述べる．提案手法はダイナミックプログラミングに基づいており，一種類の部分木を構築するのではなく，複数の部分木を構築し，得られた木に対してバッファ挿入，配線幅拡大を考慮し，最良の配線木の構築を行なう．本手法では部分木の生成に文献 [1] で提案されている $bottom\_up(T(v))$ を用いている．図 5に手続き *bottom_up* の流れを示す．この手続きは，ある配線経路について，配線経路上の可能な位置に対しバッファ挿入，配線幅拡大などを行なうものである．*bottom_up* では，節点集合の中から 2 個の節点を選択しそれぞれの節点を根とする部分木について，遅延スラック q，*dc_connect* 部分木のキャパシタンス $c(T(v))$，総配線キャパシタンス *c_total*の 3 項組 $(q(v), c\_total(T(v)), c(T(v)))$ を計算する．3 項組の計算後，選択した 2 個の節点をマージする．次にマージして得られた節点を根とする部分木に対しバッファ挿入，配線幅拡大を行ないそれぞれについて 3 項組を計算する．*bottom_up* で得られた 3 項組はアルゴリズムが終了するまで常に保持するものとする．

次に提案手法の概要について述べる．

i 個のシンクを持つ部分木の根の集合を TV_iと定義する．提案手法では配線木はシンクから節点を組み合わせていきソースまでボトムアップに構築していく．まず，シンク自身が部分木となる部分木の根の集合TV_1を構成する．次に部分木に含まれるシンク数が 2 の部分木の根の集合TV_2を構成する．この操作をシンク数回繰り返す．根集合の作成方法について説明する．根集合TV_hの要素を v_{hj}, TV_kの要素を v_{kl}とし，$1 \leq h, k \leq i$ とすると根集合 TV_iは $i = h + k$なる TV_h, TV_kの各根集合より節点ペア (v_{hj}, v_{kl}) をつくり，この節点対をマージした節点を TV_iの要素とする (図 7)．節点ペア数は組合せ数が膨大になるのを防ぐため，1 つの節点につき 2 つの節点ペアを作るものとする (図 6)．

以上の操作を行なう関数を $Find()$ とする．$Find()$ で求めた v_{im} を根とする部分木 $T(v_{im})$ に対し 3 項組を計算するために，$bottom\_up(T(v_{im}))$ を適用してバッファ挿入，配線幅拡大を考慮した部分木を作成する．生成した部分木 $T(v_{im})$ のそれぞれの場合において遅延スラックの値 $q(T(v_{im}))$ を計算しその値をアルゴリズムが終了するまで保持する．この一連の操作を i がシンク数 n になるまで続け，最後に TV_nの中から遅延スラックが最大で総配線キャパシタンスの最小となるノードを 1 つ選択し最良の配線木を見つける．図 8に提案アルゴリズム *IWBA(Improved-WBA-tree)* の流れを示す．

```
Procedure bottom_up
 foreach v ∈ 節点集合
  if vがシンク
   シンクの3項組の集合 Z_v を計算;
  else
   z_u←左の子 u の3項組;
   z_w←右の子 wの3項組;
   Z'_v← z_u, z_w をマージした
        3項組 (q_z, c_total, c_z);
   Z'_v← Z'_v ∪ {z'_v にバッファを挿入した
        3項組 };
   (q_z − D_buff, c_b, c_total + c_b)
  end if
  z_v←φ;
  for z ∈ Z'_v
   foreach wire_width
    Z_v← Z_v ∪ {q_z − D_wire(e_v),
         c_z + c_e_v, c_total + c_e_v};
    /*可能な配線幅を e_v に割り当てる*/
   end foreach
  end for
 end foreach
```

図 5: 手続き bottom_up

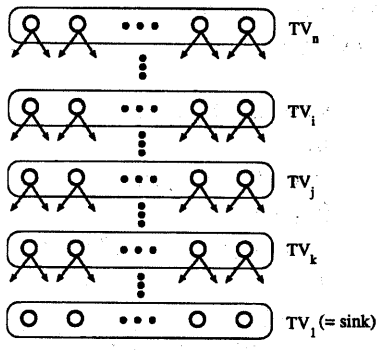

図 6: 部分木構成法

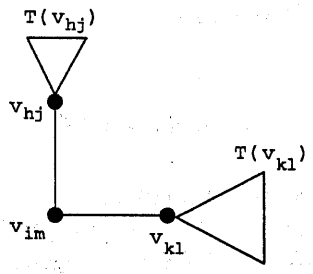

図 7: 節点のマージ

3.4 節点ペアの選択

本節では $Find()$ での節点ペアの選択方法について述べる．提案手法では節点ペアを生成する際にまず，節点ペアの組合せ候補となる部分木の根集合 TV_j, TV_k に含まれている各節点について，その節点がタイミングがクリティカルかどうかを判断する．注目している節点がクリティカルな節点かそうでないかによって，アルゴリズムの動作が異なる．

(1) クリティカルな節点の場合

注目している節点の遅延スラックが要求遅延時間の p% (p は定数) 以下であれば，その節点はクリティカルな節点であると判断する．クリティカルな節点は，現在のレベル以降での節点ペアの組合せを禁止する事を示すフラグを立てる．ここで組合せ禁止フラグを立てられた節点については，フラグが解除されるまで節点ペアを構成する事ができない．組合せ禁止フラグを解除するためには以下の条件が必要となる．まず，ソースの片方の枝を注目している節点に接続し，もう片方の枝には，注目している節点が根となる部分木に含まれる節点を除く残りの節点より構成した部分木を接続する．この部分木の構成のアルゴリズムは *A-tree* のアルゴリズムを用いて生成する．この様にして得られた配線木を仮配線木としてソースの遅延スラックを見積もる配線木として利用する．仮配線木のソースの遅延スラックを計算しその値が一定値 *const* 以上であれば，注目している節点の組合せ禁止フラグを解除する事が出来る．この様な操作を行なうことにより，ソースからクリティカルな節点までのパス上にある内部節点の数を減少させることができ，クリティカルな節点までのスラックに余裕を持たせることが出来る．

(2) クリティカルでない節点の場合

注目している節点のタイミングがクリティカルでない場合，その節点から距離の近い節点5つを選択し，その節点のなかで注目している節点と組み合わせた場合の遅延スラックが最も大きいものから順に2つの節点

```
Algorithm IWBA
 for(1 ≤ i ≤ n)
   if i = 1
    bottom_up(TV_i);
   else
    foreach vertex pair
    (v_hj, v_kl)
     = Find(TV_1, ∪···∪, TV_{i-1});
    do
    T(v_im)
     = Marge(T(v_hj), T(v_kl));
    bottom_up(v_im);
 end for
 v_best = Find_best(TV_n);
 Backtrace(v_best);
```

図 8: 提案アルゴリズム

を選択し，節点ペアを生成する．

以上 (1)(2) の操作をアルゴリズムが終了するまで繰り返す．節点ペア生成手続き *Pair_Selection* の流れを図 9に示す．ペア候補となる節点の含まれる部分木の根集合を TV_j, TV_kとし，注目している節点を v_c, v_cのペアとなる節点を v_a, v_bとする．

4 アルゴリズムの拡張

提案手法では，最終的な配線トポロジを決定するまでに，ダイナミックプログラミングにより複数の部分木を構築するため，部分木の組合せ数が膨大な数になる．実用的な計算時間で計算を行なうため木の構築過程において効果的に枝刈りを行なう必要がある．本手法では，最終的な解の質に影響を及ぼさない枝刈り手法として文献 [1] で提案されている *redundant_delete* を用いると共に，概略配線問題をいくつかの部分問題に分割し，階層的に配線木を構築する手法 *Area_divide* を提案する．

4.1 手続き redundant_delete[1]

節点 vを根とする部分木を T_v，その部分木の総配線キャパシタンスを，$c_{total}(T(v))$，vに直接接続されている枝 (*dc_connect*) のキャパシタンスを $c(T(v))$ とすると，vは $(q(v), c(T(v)), c_{total}(T(v)))$ の 3 項組を持っている．部分木の構築の過程で，ある部分木の根集合に含まれる任意の 2 つ節点の 3 項組 (q, c, c_{total})，(q', c', c'_{total})

```
Procedure Pair_Selection
 foreach v_c ∈ {TV_j} ∪ {TV_k}
  /* クリティカルな節点の判断 */
  if (v_cのスラック) <= RAT(s_i) の p %
   /* v_cはクリティカル */
   /* v_cに禁止フラグを立てる */
   flag(v_c) = 1;
  else
   /* v_cはクリティカルでない */
   flag(v_c) = 0;
  end if
  if flag(v_c) == 1
   /* フラグ解除条件 */
   仮配線木を作る;
   if ソースのスラック >= const;
    flag(v_c) = 0;
   end if
  else if flag(v_c) == 0
  /* 節点ペア生成 */
   T_temp ← {TV_j} ∪ {TV_k} から
            v_cに近いノード 5 個選択;
   /* スラックの小さい節点 2 個 */
   v_a, v_b ∈ T_temp を選択;
   (v_c, v_a), (v_c, v_b); /* 節点ペア */
  end if
 end foreach
```

図 9: 節点ペアの選択 Pair_Selection

について

$$c \leq c',\ c_{total} \leq c'_{total},\ q \geq q', \tag{7}$$

が成り立つ時，3 項組 (c', q', c'_{total}) をもつ部分木は，現在のレベル以降のアルゴリズムの実行過程で不要である事が文献 [1] で示されているので，この 3 項組をもつ節点は部分木の根集合より削除する．また，木の構築過程で，

$$q \leq 0 \tag{8}$$

の条件を満たす部分木，つまり，タイミング制約を違反している部分木も同様に削除する．

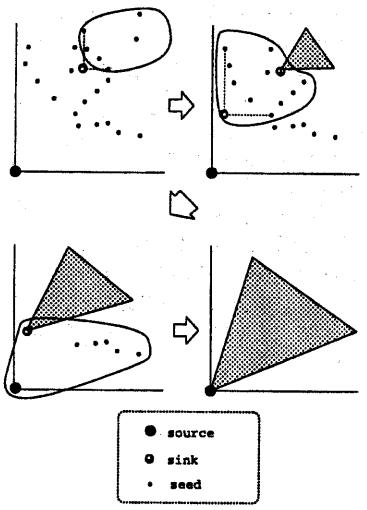

図 10: 手続き Area_divide の動作

4.2 手続き Area_divide

Area_divide では，与えられたシンク集合に対して，以下に示す指標により提案アルゴリズム IWBA を適用する範囲を分割する．分割された領域のシンクの中で x, y座標の最小値を持つノードをシードとして IWBA を適用し，その範囲でシードを根とした部分木を構築する．分割した領域内にあるシンクをシンク集合から削除し，新たにシンク集合にシードを付け加え，同様に部分木を構築する．最終的にシンク集合が空集合になるまでこの手法を適用し，最終的な配線トポロジを得る．手続き *Area_Divide* の流れを図 11に示す．

シンク数 m のシンク集合を $S_m = \{s_1, ..., s_m\}$ とし，シンク $s_i \in S_m$ の座標を $(x(s_i), y(s_i))$ とする．分割された領域の数を kとし，各領域のシンク集合を $A_i(1 \leq i \leq k), S_m = A_1 \cup, A_2 \cup, \cdots, \cup A_k$とする．

手続き *Area_Divide*() では，最初にシンク集合の中からソースより最も距離の遠いシンク s_iを 1 つ選択する．その s_iからの距離が近いシンクを l個 (l は定数) 選択し集合 A_iに含める．次に選択されたシンク集合 A_iをシンク集合 S_mより取り除く．領域分割数が k個になるまで以上の操作を繰り返す．提案手法では手続き *Area_Divide* で得られた各シンク集合 A_iに対して *IWBA* を適用することにより，階層的に問題を解くものとする．

Procedure *Area_Divide*
While $|S_m| - \sum |A_i| \neq 0$
最も遠いシンク s_iを S_mより見つける;
if $|A_i| < k$
$A_i \leftarrow s_i$から近いシンク l個;
$S_m \leftarrow S_m - \{A_i\}$;
end if
$i++$;
end while

図 11: 手続き Area_Divide

5 実験的評価

提案手法を C 言語を用いて Ultra COMP station model 170 上に実現しシミュレーション実験を行なった．実験に用いたデータは，10mm × 10mm の配線領域において，グリッド幅を $100\mu m$ としたランダムデータを用いた．実験ではシンク数 10, 20, 30, 50 のシンク位置と各シンクの要求遅延時間が異なる 3 種類のデータを生成し提案アルゴリズムを適用した．実験に用いた物理定数は単位長，単位面積あたりの配線キャパシタンスを $0.04(fF/\mu m)$，単位長あたりのフリンジキャパシタンスを $0.15(fF/\mu m)$，単位長あたりの抵抗を $0.12(\Omega/\mu m)$，バッファの固有遅延を $0.125(ns/\mu m)$，バッファ抵抗を $814(\Omega)$，バッファのロードを $0.028(ns/\mu m)$ となっている [1]. 実験結果を表 1 に示す．表 1 では左のコラムから順に実験に用いたネットのシンク数，バッファ数制約，ソースでのスラック，総配線キャパシタンス，計算時間を表している．上記結果より，バッファ数制約

表 1: 実験結果

シンク数/バッファ数	$q(s_0)[ps]$	$C(s_0)[pF]$	時間 $[ms]$
10 / 0	5577	25.32	70
10 / 2	6958	27.11	80
20 / 0	1526	59.19	170
20 / 2	4688	59.34	2390
30 / 0	4065	74.46	260
30 / 2	4728	76.80	3360
50 / 0	4703	126.48	480
50 / 2	4955	125.608	6080

が 0 と 2 の場合ではどのデータでもバッファを入れたデータの方が遅延スラックは大きくなった．同じシンク数のデータであっても計算時間がこの様に異なってくる理由として，部分木の枝刈がシンクの座標位置に

よって，多く出来る場合とそうでない場合があるためであると考えられる．

6 あとがき

今後の課題としては，節点ペアの組合せ数削減のための効率の良い枝刈り手法の開発，領域分割における新たな分割指標提案，複数の手法を融合した領域分割指標を開発，及び大規模データに対して提案手法を適用する事などが挙げられる．更に，提案手法と文献 [1] および他の手法との比較実験を行なう予定である．

謝辞

提案手法を実現するにあたり，文献 [1] の手法について多くの御教示を賜わりました NEC C&C メディア研究所の岡本匠氏に深謝致します．

参考文献

[1] T. Okamoto and J. Cong: "Buffered Steiner tree construction with wire sizing for interconnect layout optimization," Proc. of ICCAD, pp. 524–527 (1996).

[2] J. Luis, C.-K. Cheng and T. T. Y. Lin: "Simultaneous routing and buffer insertion for high performance interconnect," Proc. of Fifth ACM/SIGDA Physical Design Workshop, pp. 7–12 (1996).

[3] C. Alpert and A. Devgan: "Wire segmenting for improved buffer insertion," Proc. DAC, pp.588–593 (1997).

[4] J. Cong, l. He, C. K. Koh and P. H. Madden: "Performance optimization of VLSI interconnect layout," INTEGRATION, the VLSI Journal, Vol.21, pp.1–94 (1996).

[5] J. Lillis, C. K. Cheng and T. T. Y. Lin: "Optimal wire sizing and buffer insertion for low power and a generalized delay model," Proc. ICCAD, pp.138–143 (1995).

[6] J. O. I. Pyo and M. Pedram: "Constructing lower and upper bounded delay routing trees using linear programming," Proc. DAC, pp.508–512 (1996).

[7] Y. Y. Li and S. K. Cheung et al. : "On the Steiner tree problem in $\lambda 3$-metric," Proc. ISCAS, pp.1564-1568 (1997).

[8] S. K. Rao, P. Sadayappan, F. K. Hwang and P. W. Shor: "The rectilinear Steiner arborescenes problem," Algorithmica, Vol.7, pp.277–288 (1992).

[9] S. D. Sapatnekar: "RC interconnect optimization under the Elmore delay model," Proc. DAC, pp.387–391 (1992).

[10] L. N. Kannan et al. :"A methodology and algorithms for post-placement delay optimization" Proc. DAC, pp.327–332 (1994).

[11] K. S. Leung and J. Cong: "Fast optimal algorithm for the minimum rectilinear Steiner arborescence problem," Proc. ISCAS, pp.1568–1571 (1997).

[12] 大釜, 小出, 若林: "タイミング制約を考慮した概略配線手法" 情報処理学会全国大会講演論文集, Vol.1, pp.123–124 (1997).

[13] 三林, 高橋, 梶谷: "総長の総和と最大に関する均衡平面スタイナー木", DA 研究報告 85-7, pp.37–44 (1997).