設 計 自 動 化　43－3
（１９８８．７．１２）

フロアプランにおける配線経路指定の一手法

山内貴行、岡田時仁、三浦順子、神戸尚志
シャープ株式会社　コンピュータシステム研究所

本論文では、VLSIのフロアプラン設計における配線経路指定問題についての検討をおこなう。本文では、まずフロアプランにおける配線経路に対する考慮の重要性を述べる。次に経路指定機能を実現する際に、ブロックの位置や形状の変化がもたらす問題点をあげる。そして、フロアプラングラフとそれを用いた配線経路の表現について説明する。さらに、ブロックの移動量の分割とタイル処理にもとづく、フロアプラングラフの局所的更新手法を提案する。そして、ブロック移動前のフロアプラングラフを用いて表現された経路を、移動後のグラフを用いて再表現することにより、配線経路が維持できることを示す。

A Method to Keep Global Route for Specified Nets on Floorplanning

Takayuki Yamanouchi, Tokihito Okada, Junko Miura, Takashi Kambe
Computer Systems Laboratories, SHARP Corporation
2613-1 ICHINOMOTO-CHO, TENRI NARA 632 JAPAN

In this paper, we study about the method to keep global route for specified nets on VLSI Floorplanning design. It is important to consider grobal route for some nets on floorplanning. We discuss the problem caused by the change of the position or the shape of the blocks, when the global route is guided. We propose the local updating method of Floor Plan Graph, based on gradually movement of the blocks and processing tiles.
Then, it is shown that the global route is kept by re-representing global route with Floor Plan Graph.

1. はじめに

1.1 フロアプラン設計

LSIの大規模化に伴い、そのレイアウトは複数の機能ブロックを組み合せてチップを構成する階層的な手法が採られる場合が多くなっている。機能ブロック内のレイアウトには、過去の設計資産や、モジュールコンパイラ[1]、スタンダードセル方式レイアウト[2]等の設計自動化ツールを用いて効率よく設計をおこなうことができる。しかしながら、チップ全体の最適化を図るためには、機能ブロック内のレイアウトをおこなう以前に、各ブロックの働きや、ブロック間･外部との接続を考慮して、レイアウトの方針を大局的に決定するフロアプラン設計が重要である。

フロアプランシステムは、フロアプラン設計の自動化を図り、チップ全体の最適化の効率を高めるためのもので、当社で開発中のシステムでは

- ･未設計ブロックの面積見積り
- ･ブロックの自動配置[3]
- ･未設計ブロックの形状･端子位置の自動決定
- ･ブロック間の配線領域の見積り
- ･配線領域見積りに応じたブロック位置の最適化
- ･配線の電気的特性の指定および予想
- ･自動および人手での配線経路の指定

等の機能を持つ。特に、アナログ信号線や電源線のような電気的特性に対して厳しい仕様を持つ配線については、配線長のみならず配線経路もチップの性能に大きな影響をあたえるため、フロアプランの際にその経路を検討･指定する必要がある。配線経路に対する考慮は、1チップの規模が大きくなり、また、1チップ内で実現される機能が複雑になる程重要になる。

1.2 配線経路指定の問題点

フロアプランの途中で指定された配線経路はレイアウト結果に反映されなければならない。しかし、ブロックの位置や形状は、経路指定後も不変ではないため、変化する配線領域に応じて指定された経路の表現を変えていく必要がある。配線経路指定のために配線領域固定とすれば、未配線が生じたり、面積や配線長の最適化が困難となり、経路表現が配線領域の変化に対応できなければ、経路指定は大半の配線で実現できなくなる。

例えば、配線領域の見積りは、経路指定された配線も含めておこなう必要があるが、見積りの結果をもとに、必要な配線領域を確保したり無効領域を除くために、自動あるいは人手でブロックの位置を動かす必要がある(図1-A)。また、ボトムアップにレイアウトをおこなった結果、ブロックの形状がフロアプラン時と若干変わる(図1-B)。このようにブロックが移動や変化した場合には、指定された配線経路を維持する必要がある。この場合、絶対位置による経路表現は、ブロック位置の変化に対処できない。従って、配線経路の表現は、ブロックの相対位置関係とともに表現する必要がある。

本論文では、フロアプライングラフ[4]とそれを用いた配線経路表現について説明し、ブロックの移動に伴うフロアプライングラフの変化を、タイルの処理にもとづくグラフの更新によっておこなう手法を提案する。そして、配線経路の維持は更新後のフロアプライングラフにもとづいて経路を再表現することにより、実現できることを示す。

2. フロアプライングラフと配線経路表現

2.1 フロアプライングラフ

本文では、ブロックの相対位置や、配線領域、

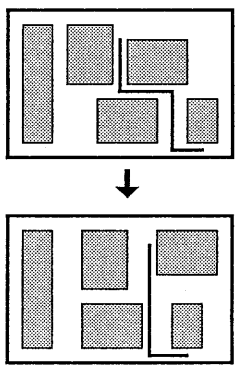


図1-A

配線領域の見積りによるブロック位置の移動

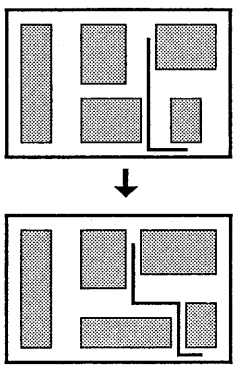


図1-B

下位ブロックのレイアウト結果による形状の変更

図1 ブロック位置や形状の変化に伴う配線領域の変更

配線経路を表現するモデルとしてフロアプラングラフ[4]を用いる。配線経路の維持は、フロアプラングラフを用いておこなわれる。また、未設計ブロックの端子位置決定や、配線領域の見積り、配線領域の見積りにもとづくブロック位置の自動修正などの処理は、フロアプラングラフ上での概略配線をもとにおこなわれる。自動レイアウトツールへフロアプラン結果の情報を引き渡す際にもフロアプラングラフが用いられる。

ここで、本文で取り扱うレイアウトモデルを説明する。

- ブロックは矩形であり、ブロックの周辺上には、接続用の端子が存在する。
- フロアプランをおこなう領域を表す矩形が存在し、すべてのブロックは、その矩形内部に重なることも接することもなく配置されている。
- ブロックの移動は、ブロックの相対配置や形状可変ブロックの形状がほぼ決定され、配線経路指定がおこなわれた後の、小規模なものを取り扱う。複数のブロック位置の交換をおこなうような移動があった場合は配線経路指定も無意味なものと考えられるためである。

次にフロアプラングラフについて、タイルの概念を用いて説明する。

フロアプラン領域内の部分領域を示す矩形をタイルと呼ぶ。ブロックの領域をブロックタイルと呼び、ブロック外の領域を非ブロックタイルと呼ぶ。非ブロックタイルには、フロアプラン領域を水平方向に分割(水平分割と呼ぶ)して作られる水平非ブロックタイルと、垂直方向に分割(垂直分割と呼ぶ)して作られる垂直非ブロックタイルがある。さらに、上下[左右]の辺がともにブロックタイルに接する水平[垂直]非ブロックタイルを水平[垂直]ドミナントタイルと呼び、それ以外の水平[垂直]非ブロックタイルを水平[垂直]スペースタイルと呼ぶ。

フロアプラングラフは、次のようにして得られる。

(1) フロアプラン領域をブロックタイルと、水平非ブロックタイルに水平分割する[5]。このとき、水平非ブロックタイルは水平方向にできる限り長くなるように分割する。すなわち、水平非ブロックタイルの左右の辺に水平非ブロックタイルが接することがないように分割する。

(2) 水平非ブロックタイルを水平ドミナントタイルと水平スペースタイルに分類する。

(3) 同様にして、フロアプラン領域を垂直分割し、垂直非ブロックタイルを分類する。(図2)。

(4) 水平ドミナントタイルと垂直ドミナントタイルを重ね合わせた時、両者のドミナントタイルが重複する部分に対して頂点を対応させ、ドミナントタイルが重複しない部分に対して辺を対応させることによって得られるグラフをフロアプラングラフとする(図3)。

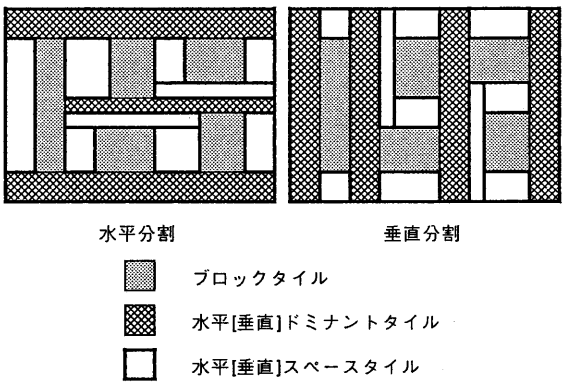


図2 領域の水平·垂直分割

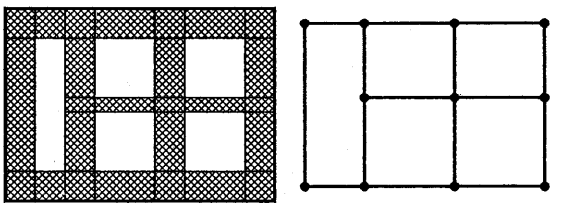


図3 フロアプラングラフの生成

2.2 フロアプラングラフによる配線経路表現

指定された配線経路はブロック周辺上の端子と、フロアプラングラフの頂点と辺の集合によって表す(図4)。グラフの辺はブロック間の相対的な配線領域を表しており、配線経路も、相対的に表現されたことになる。

3. フロアプラングラフの更新による配線経路の維持

3.1 ブロック移動量の分割

ブロックの位置に変化が生じた場合、フロアプラングラフは変化の前後で異なる場合が生ずる(図5)。従って、変化前のグラフをもとに表現されてい

る配線経路は、変化後のグラフをもとに表現し直す必要がある。複数のブロックの位置に変化が生じた場合、グラフの辺や頂点の消滅、生成が起きるために、変化の前後で得られるグラフのみを比較して経路の再表現をおこなうことは難しい問題となる。そこで、すべての変化を一つのブロックの一次元方向の移動の組合せに分解し、さらに一回の移動量をグラフの局所的変更が可能な、拡大･縮小に分解すること(図6)によって、フロアプライングラフを徐々に更新する手法を検討した。

[補題1] フロアプライングラフの辺は、ドミナントタイルの各部分との対応関係が成り立つため、ブロックタイルの移動によって生ずるドミナントタイルの生成、消滅が起こる度に、フロアプライングラフの対応する部分に辺や頂点の追加･除去をおこなえばよい。

[補題2] ブロックタイルの水平[垂直]方向の移動は、フロアプラン領域を垂直[水平]分割したタイルの相対位置を変えないため、タイルの生成、消滅は垂直(水平)分割したタイルについて考慮すればよい。

[補題3] ブロックの変化は、ブロックの一辺の拡大操作と縮小操作の組合せによって実現できるため、タイルの変化は、ブロックの一辺の一次元方向の拡大と縮小について考慮すれば充分である。

補題1,2,3 にもとづき、以下に、ブロックタイルの左辺のX軸方向の拡大と縮小の場合についてタイルに対する処理を示す。

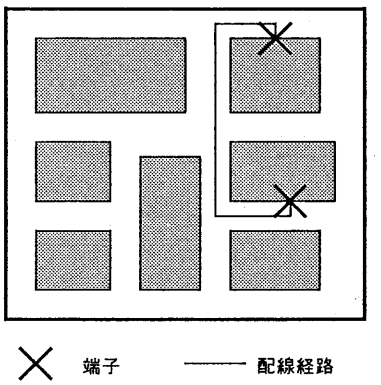
3.2 グラフ更新のためのタイル処理

拡大または縮小するブロックタイルをBTと呼ぶ。

BTの左辺に接するタイルをLTと呼ぶ。(LTは非ブロックタイルで、一つしか存在しない)

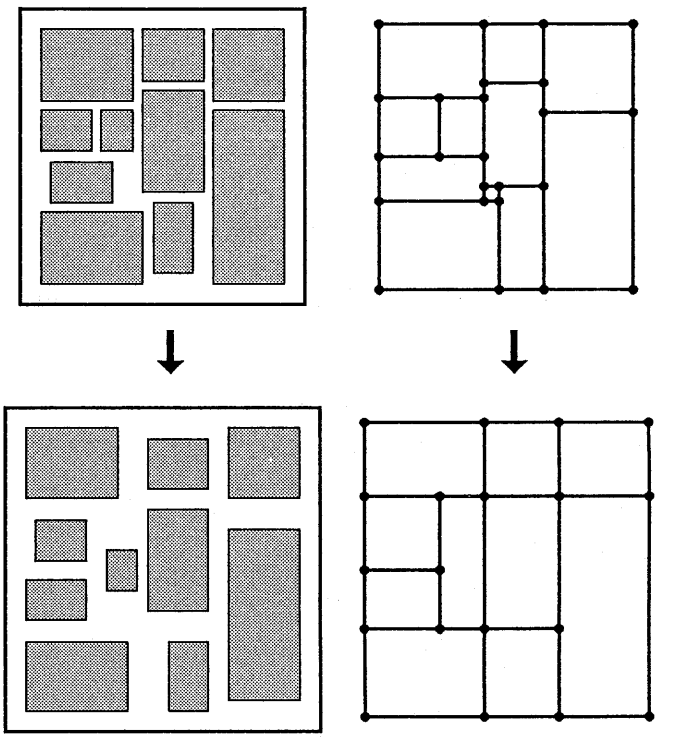
BTの上辺に接するタイルのうち、最も左側にあるタイルをUTとする。

BTの下辺に接するタイルのうち、最も左側にあるタイルをDTとする。



端子とグラフの辺･頂点(太線で示す)
の集合によって経路を表現する

図4 フロアプライングラフによる経路表現



ブロックの見た目の相対位置はほぼ同じだが、
フロアプライングラフは全く異なる

図5 ブロックの移動による
フロアプライングラフの差異

BT,LT,UT,DTの左下、右上の座標をそれぞれ

BT(btx1,bty1), (btx2,bty2)

LT(ltx1,lty1),(ltx2,lty2)

UT(utx1,u1y1),(utx2,u1y2)

DT(dtx1,dty1),(dtx2,dty2)

とする(図7)。

タイル処理の概要は次の通りである。

[補題4] BTの1回の拡大量をLTの幅に限定すれば、タイルの構成に影響がおよぶのは、BT,LT,UT,DTの占める部分領域に限定できる

[補題5] BTの1回の縮小量をUT,DTの幅に限定すれば、タイルの構成に影響がおよぶのは、BT,LT,UT,DTの占める部分領域に限定できる

補題4,5に従いタイル処理をおこなう。

BTが拡大する場合、LTが縮小し、UTとDTは、BT

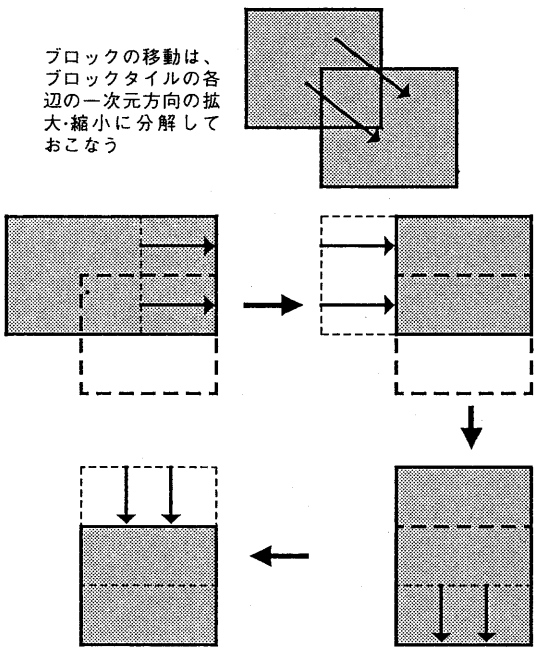


図6 ブロック移動の分解

(ブロック拡大処理)

```
PROCEDURE EXPAND_BLOCK (BT, Δ)
      /* BTの左辺をX軸負方向にΔ拡大する */

WHILE (Δ>0) DO

   Wd = ltx2 - ltx1 ;
   IF (Δ<Wd)
      δ=Δ ; Δ=0 ;
      ltx2 = ltx2 - δ ; btx1 = btx1 - δ ;

      IF (uty2 = lty2)
         utx1 = utx1 - δ ;
      ELSE
         INSERT_TILE( S1(btx1,bty2), (utx1,lty2) )
         SET_TYPE( S1, "スペースタイル" )

      IF (dty1 = lty1)
         dtx1 = dtx1 - δ ;
      ELSE
         INSERT_TILE( S2(btx1,lty1), (dtx1,bty1) )
         SET_TYPE( S2, "スペースタイル" )

   ELSE IF (Δ≧Wd)
      δ = Wd ;, Δ = Δ - δ ;
      DIVIDE_TILE(LT(ltx1,lty1),(ltx2,lty2),LT1,
                  LT2,LT3, bty2,bty1 )
      DELETE_TILE( LT2 )
      btx1 = btx1 - δ ;

      [BTの左辺に新たに接するタイルをTAとする]
      JUDGE_TYPE( TA )

      IF (uty2 = lty2)
         MERGE_TILE( UT, LT1 )
         JUDGE_TYPE( UT )
      ELSE
         JUDGE_TYPE( LT1 )

      IF (dty1 = lty1)
         MERGE_TILE( DT, LT3 )
         JUDGE_TYPE( DT )
      ELSE
         JUDGE_TYPE( LT3 )

END
```

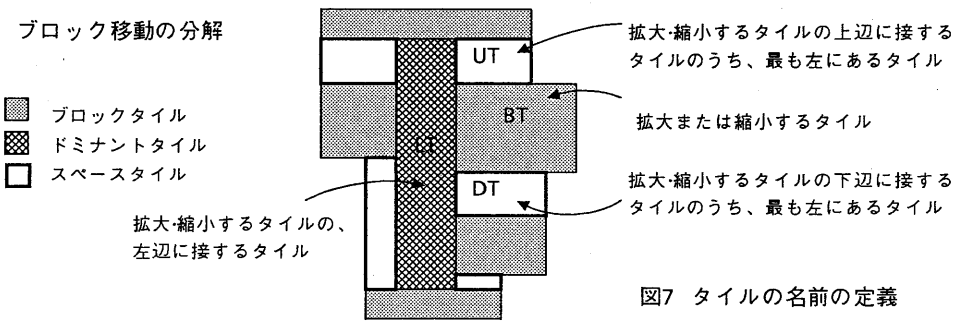


図7 タイルの名前の定義

(ブロック縮小処理)

```
PROCEDURE SHRINK_BLOCK (BT, Δ)
  /* BTの左辺をX軸正方向にΔ縮小する */

WHILE (Δ>0) DO
  Wd = min( utx2 - utx1, dtx2 - dtx1 ) ;
  IF (Δ<Wd)
    δ = Δ ;, Δ = 0 ;
    btx1 = btx1 + δ ;
    utx1 = utx1 + δ ; dtx1 = dtx1 + δ ;
    IF (uty2 = lty2 AND dty1 = lty1)
      ltx2 = ltx2 + δ ;
    ELSE
      INSERT_TILE( IT(ltx2,dty1), (btx1,uty2) )
      JUDGE_TYPE( IT )
      JUDGE_TYPE( LT )
      JUDGE_TYPE( UT )
      JUDGE_TYPE( DT )
  ELSE IF (Δ≧Wd)
    flag = FALSE ;
    δ = Wd ;, Δ = Δ - δ ;
    btx1 = btx1 + δ ;
    utx1 = utx1 + δ ; dtx1 = dtx1 + δ ;
    IF (uty2 = lty2 AND dty1 = lty1)
      ltx2 = ltx2 + δ ;
    ELSE
      INSERT_TILE( IT(ltx2,dty1), (btx1,uty2) )
      flag = TRUE ;
    IF (utx1 = utx2)
      DELETE_TILE( UT )
    ELSE
      JUDGE_TYPE( UT )
    IF (dtx1 = dtx2)
      DELETE_TILE( DT )
    ELSE
      JUDGE_TYPE( DT )
    IF ( flag = TRUE )
      JUDGE_TYPE( IT )
    JUDGE_TYPE( LT )
END
```

(タイルに対する補助処理)

```
PROCEDURE INSERT_TILE ( T(x1,y1),(x2,y2) )
      /* 左下、右上の座標を(x1,y1),(x2,y2)とする
      タイルTを挿入する */

PROCEDURE DELETE_TILE ( T )
      /* タイルTを削除する */

PROCEDURE MERGE_TILE ( T1, T2 )
      /* タイルT1とT2を合成し、T1とする */

PROCEDURE DIVIDE_TILE ( T(x1,y1),(x2,y2), T1, T2,
                        T3, ya, yb )
      /* タイルTをT1,T2,T3に分割する */

  DELETE_TILE( T )
  INSERT_TILE( T1(x1,ya),(x2,y2) )
  INSERT_TILE( T2(x1,yb),(x2,ya) )
  INSERT_TILE( T3(x1,y1),(x2,yb) )

END

PROCEDURE SET_TYPE ( T, TYPE )
      /* タイルTの種類をTYPEに設定する */

PROCEDURE JUDGE_TYPE ( T )
      /* タイルTがドミナントタイルかスペース
      タイルかを判定する */
```

とともに拡大するか、あるいは、新たなタイルが挿入される。

BTが縮小する場合、UTとDTはBTとともに縮小し、LTはBTとともに拡大するか、あるいは、新たなタイルが挿入される。

詳細な処理は*PROCEDURE*として記す。なお、ブロック拡大処理の妥当性を付録に載せておく。

3.3 タイル処理にもとづくフロアプライングラフの局所的更新

[補題6] フロアプライングラフを更新する必要があるのは、*PROCEDURE* JUDGE_TYPEでスペースタ

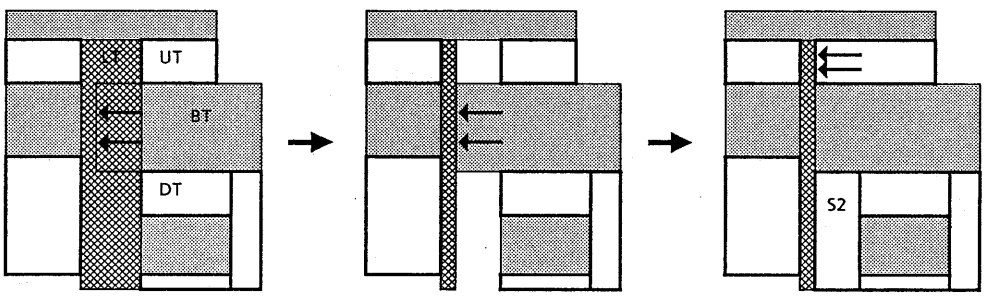


図8.1 タイル拡大処理例1

グラフに対する影響なし

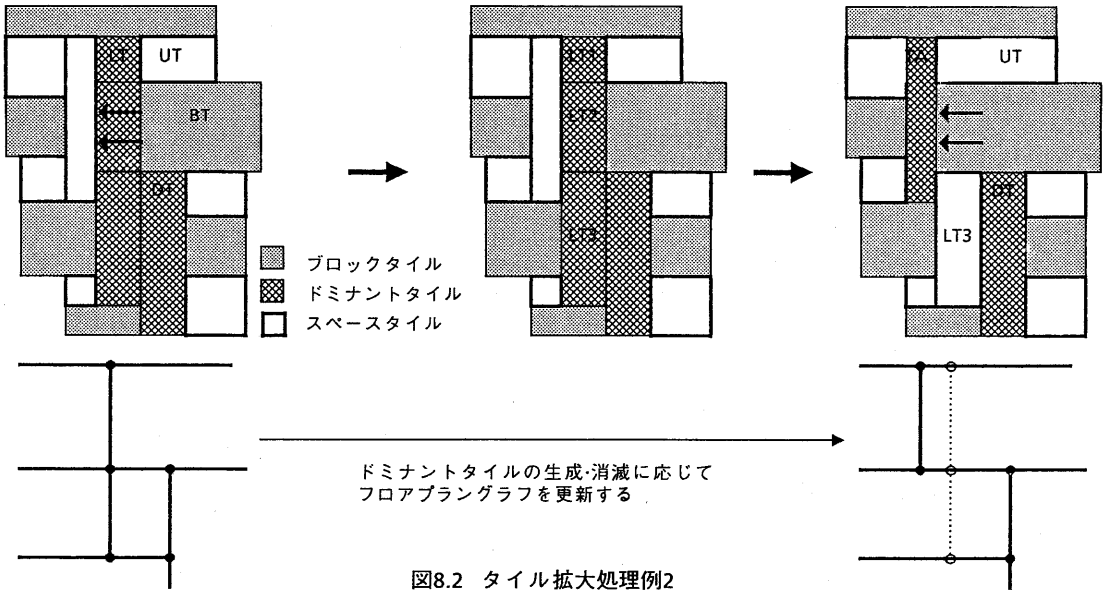


図8.2 タイル拡大処理例2

イルがドミナントタイルに変わる、あるいは、ドミナントタイルがスペースタイルに変わる場合と、*PROCEDURE* DELETE_TILEでドミナントタイルが消滅する場合である(図8.1)。

スペースタイルがドミナントタイルに変わる場合とドミナントタイルが消滅する場合は、消滅する部分に対応するフロアプライングラフの辺を消去する。このとき、消去する辺に接続する頂点について、消去する辺以外に垂直方向の辺と接続しない場合、その頂点に接続する水平方向の辺を一つにまとめ、頂点を消去する。

スペースタイルがドミナントタイルに変わる場合、新しく生成したドミナントタイルと、水平ドミナントタイルが交差する箇所に対応する水平方向の辺を二つにわけ、新しく頂点を生成し(既に頂点が存在する場合はそのまま)、生成された頂点間を結ぶ辺を追加すればよい(図8.2)。

3.4 グラフの更新に伴う経路の再表現

タイルの拡大･縮小処理によって、ブロックの拡大･縮小に伴いフロアプライングラフが変化する際に、消滅または生成される辺や頂点はすべて認識することができる。同様に、変化の前後で影響を受けない辺や頂点の、二つのグラフ間の対応関係をとることができる。よって、グラフの更新に伴う配線経路の再表現は、容易におこなうことが可能である。

グラフの変化後、消滅する辺や頂点を経路集合の中に持つ配線経路についてのみ経路の再表現をおこなう。

経路の消滅した部分の補充は、単に消滅する経路の両端を更新後のグラフの最短経路で結ぶのでは不十分である。消滅する辺がグラフの更新前に対応していたドミナントタイルの両側に接していたブロ

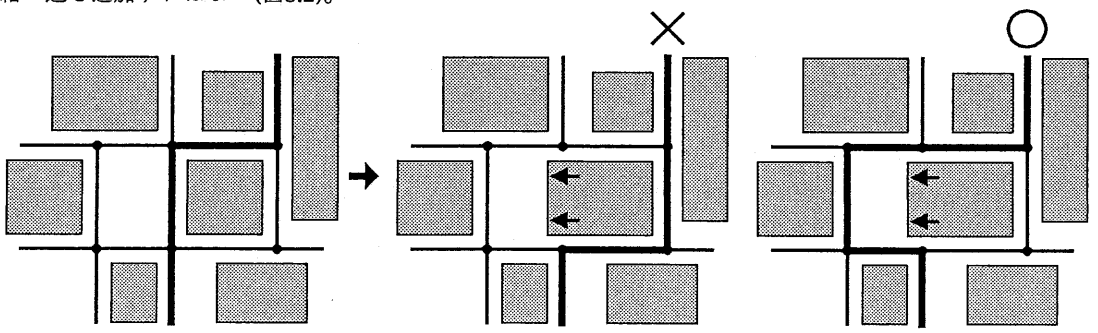


図9 経路の再表現

ック間に、グラフの更新後存在するドミナントタイルと対応する辺を必ず通過するように、経路を補充する。該当するタイルが複数存在するときは、消滅するドミナントタイルの位置に最も近いドミナントタイルを選ぶものとする。これによって、移動に伴う経路の再表現の前後で、経路がブロックの同じ側の横を通過することを保証することができる(図9)。

4. まとめ

本論文では、フロアプランにおける配線経路指定の問題点をあげ、フロアプライングラフにもとづく配線経路の表現方法を提案した。さらに、ブロックの位置や形状変更の際におこるフロアプライングラフの変化に対し、移動量の分割とタイルの移動による、フロアプライングラフの局所的更新手法を提案し、グラフの対応関係による配線経路の再表現方法について述べた。

本手法は、ブロック位置の小規模な移動を前提としており、ブロック位置の大規模な変動の際には、意味が無くなったと考えられる経路まで、経路の再表現によってレイアウトしてしまう恐れがある。

我々は、フロアプライングラフ更新のプロトタイププログラムを開発し、ブロック位置の小規模な移動について、本手法の有効性を確認した。さらに、どの程度のブロックの移動まで妥当な経路が維持されるか現在評価中である。

参考文献

[1]原嶋、他:C言語によるレイアウト記述の一手法、信学技報、CAS87-14、(1987-5)

[2]神戸、他:ビルディングブロック型スタンダードセル方式LSIの自動レイアウトシステムSHARPSIIについて、信学技報、CAS85-141、(1986-1)

[3]冨田、神戸:ビルディングブロック方式LSIにおけるルールベースを用いた一配置手法、信学技報、VLD87-96、(1987-10)

[4]W.Dai, T.Asano, E.S.Kuh, : Routing Region Definition and Ordering Scheme for Building-Block Layout, IEEE Trans. on Computer-Aided Design, Vol. CAD-4, No3, JULY 1985

[5]J.K.Ousterhout, : Corner Stitching: A Data-structuring Technique for VLSI Layout Tools, IEEE Trans. on Computer-Aided Design, Vol. CAD-3, No1, JAN 1984

[付録] タイル拡大処理の妥当性の証明

(紙面の都合上拡大処理のみ証明を示す)

(タイル領域についての定義)
タイルの上辺･左辺は境界線上を含み、下辺･右辺は境界線上を含まない

(タイルの隣接についての定義)
タイルT1の右辺とタイルT2の左辺が接するとは
タイルT1,T2の左下、右上の座標をそれぞれ
T1(t1x1,t1y1)(t1x2,t1y2),
T2(t2x1,t2y1)(t2x2,t2y2)
とするとき、
t1y2>t2y1 *AND* t1y1<t2y2 *AND* t1x2=t2x1
なる状態であることをいう。

(証明)
BTはブロックタイルで、仮定よりブロックタイルが接することはないので、BT左辺に接するタイルは非ブロックタイルである。左辺に接するタイルが複数存在するとすれば、それらの上下辺がBTの左で接することになり、非ブロックタイルの作り方に反するのでBT左辺に接するタイルはLTのみである。
δ≦Wdであるので、LTの左側に生じる線分はないので、LT左側のタイル構造が変化することはない。
uty2=lty2のときUTとLTの上辺に接するタイルは同じブロックタイルであるので、(もし、異なるとすればそれらのブロックタイルが接することになる)UTの左辺はBTの左辺のみによって、位置が決定されたものである。従って、δ<Wdのときは BT左辺を移動するとUTの左辺も移動することとなり、δ=Wdのときは、LT1とUTを合成することとなる。
uty2≠lty2のときUTとLTの上辺に接するタイルは異なるブロックタイルである。UTの左辺は、BTの左辺のみならず、UTとLTの上辺に接するブロックタイルのうち、下辺の位置が低いほうのブロックによって位置が決定されたものである。従って、δ<Wdのときは BT左辺を移動するとLTとUTの間にタイルを挿入することになり、δ=Wdのときは、LT1とUTがそのまま残ることになる。
DTについて、UTと全く同様の議論ができる。
よって、タイルの処理は、*PROCEDURE* EXPAND_BLOCKで充分である。