設 計 自 動 化 80－3
（1996．5．24）

通信処理用FPGAを対象とした配置・概略配線同時処理手法

戸川 望　　佐藤 政生　　大附 辰夫

早稲田大学理工学部電子通信学科
〒169 東京都新宿区大久保 3-4-1
E-mail: togawa@ohtsuki.comm.waseda.ac.jp

あらまし

本稿では，通信処理用 FPGA を対象に配線混雑度の最小化を目的とした配置・概略配線同時処理手法を提案する．提案手法は，レイアウト領域および配置すべき LUT（LookUp Table）集合の階層的 2 分割を基本とする．ネットワークフローの考えを適用し最大フローによる最小カットを算出することで配線混雑度の小さくなる分割を実現する．この際，分割された LUT 集合間の結線要求は，仮想端子と呼ばれる仮想的に導入した端子によって保持される．仮想端子の並びによって概略配線径路が表される．その結果，階層分割の終了と共に配置と概略配線とが決定する．計算機実験により，提案手法の有効性を評価する．

キーワード　通信処理用 FPGA，レイアウト設計，配置，概略配線，ネットワークフロー

A Simultaneous Placement ang Global Routing Algorithm for Transport-Processing FPGAs

Nozomu TOGAWA　　Masao SATO　　Tatsuo OHTSUKI

Dept. of Electronics and Communication Engineering
Waseda University
3-4-1 Okubo, Shinjuku, Tokyo 169, Japan
E-mail: togawa@ohtsuki.comm.waseda.ac.jp

Abstract

This paper proposes a simultaneous placement and global routing algorithm for transport-processing FPGAs whose primary objective is minimizing routing congestion. The algorithm is based on hierarchical bipartition of layout regions and sets of LUTs (LookUp Tables) to be placed. It achieves bipartitioning which leads to small routing congestion by applying a network flow technique to it and computing a maximum flow and a minimum cut. If there exist connections between bipartitioned LUT sets, pairs of pseudo-terminals are introduced to preserve the connections. A sequence of pseudo-terminals represents a global route of each net. As a result, both placement of LUTs and global routing are determined when hierarchical bipartitioning procedures are finished. The experimental results for practical transport-processing circuits show its efficiency and effectiveness.

Key Words *transport-processing FPGA, layout design, placement, global routing, network flow*

1 まえがき

多岐にわたる情報源や通信プロトコルに対処するため，柔軟性を持った通信デバイスとして，通信処理に特化したFPGAチップが提案されている [9],[14],[15][1]．通信処理用FPGAは，文献 [16] に代表される一般のFPGAと異なる次の特徴を持つ．

- より柔軟な回路を構成可能にするため，論理機能を実現する最小の論理単位の粒度が細かい．
- 通信処理は信号の流れの方向性が強いため，セルの入出力端子位置が揃っている．

通信処理用FPGAを対象としたレイアウト設計を考えた際，特にその上流に位置する配置・概略配線設計を考えた際，通信処理用FPGAの持つ特徴から以下の2点に留意する必要がある．

(1) 論理粒度が細かいため，セル間を結線する配線が増化する傾向がある．100%配線を実現するためには，できるだけ配線混雑度の小さい配置・概略配線を得る必要がある．

さらに，一般にFPGA設計に不可欠な点として，以下の点に留意する必要がある．

(2) FPGAの用途が，主にシステムのラピッドプロトタイピングであることから，FPGA設計そのものが高速である必要がある．

中でも (1) は，FPGA上に回路を実現するために必須である．現在までの多くのFPGA向け配置手法は，その評価指標として仮想総配線長，カットラインを横切るネット数を基本としたものが採用されている [10]．このような評価指標は直接的に配線混雑度を評価しておらず，(1) に対する要求を満足しない．(1) に対する要求を満足するには，配置処理を実行している段階で概略配線そのものを評価する必要があると考える．すなわち，ここで求められるのは配置と概略配線とが同時に，しかも (2) の要求から高速に処理されることである．

配置・概略配線同時処理手法として，これまでにビルディングブロック [3],[8]，スタンダードセル [11]，ゲートアレイ [1] を対象としたものが発表されてきた．これらの手法はいずれもその第一の評価指標が配線長の最小化であり，配線混雑度の最小化は陽に考慮されていない．一方，FPGA向け配置・概略配線同時処理手法として，これまでに文献 [5],[6],[13] の手法が発表されてきた．文献 [5],[6] では，いずれもSA (Simulated Annealing) を基本処理としており，大規模回路の最適化には極めて多くの処理時間を要する可能性がある．文献 [13] の問題点は，入力として組合せ回路をとることにある．

本稿では，通信処理用FPGAを対象に配線混雑度の最小化を目的とした配置・概略配線同時処理手法を提案する．提案手法は，レイアウト領域および配置すべきLUT (LookUp Table) 集合の階層的2分割を基本とする．ネットワークフローの考えを適用し最大フローによる最小カットを算出することで，混雑度の小さくなる分割を実現する．この際，分割されたLUT集合間の結線要求は，仮想端子と呼ばれる仮想的に導入した端子によって保持される．仮想端子の並びによって概略配線径路を表す．その結果，階層分割の終了と共に配置と概略配線とが決定する．

本稿は次のように構成される．2章で本稿で扱う配置・概略配線問題を定式化する．3章で配線混雑度の最小化を目的とした配置および概略配線の同時処理手法を提案する．4章で計算機実験により手法の有効性を評価する．5章で本稿をまとめ，今後の課題を検討する．

[1]通信処理用FPGAとして，実際にPROTEUSと呼ばれるFPGAが開発されている [9]．

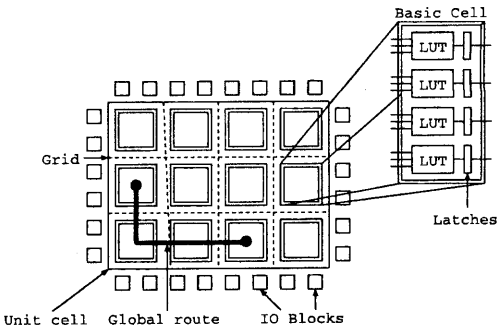

図 1: FPGA モデル

2 問題の定式化

2.1 FPGA モデル

文献 [15] で提案されたFPGAチップをもとに，図1に示すFPGAモデルを考える．FPGAモデル内部に基本セル (Basic Cell) が格子上に整列する．基本セルは，図1のように3入力1出力のLUTを縦に4つ収納する．LUTは論理を構成する最小単位である．基本セル内の各LUTは，基本セルの左辺に入力端子を3つ持ち，右辺に出力端子を1つ持つ．このような入出力端子の並びは，左から右への信号の流れを実現する．各LUTの出力はプログラムによりラッチするか否かが決定される．チップ内の全てのラッチは，特別な配線リソースによって共通のクロック信号に同期する．チップ周辺部にはIOブロックが整列する．

2.2 配置・概略配線設計

図1のFPGAモデルを対象とした配置処理は，次の (i)～(iii) の3処理から構成される．

(i) 4つのLUTをまとめて1つの基本セルを構成する，

(ii) 基本セルの配置位置を決定する，

(iii) 基本セル内部で4つのLUTの位置を決定する．

(i) は広義のテクノロジーマッピングと考えることができ，その意味で，ここでの配置問題はテクノロジーマッピングと配置とを同時に求める問題といえる．このうち，配線混雑度に大きく影響を与えるのは，(i)，(ii) の2処理である．(iii) の処理は，各基本セルの中で4つのLUTの配置位置を全通り試すことで容易に実現できる．すなわち，(iii) は，(i),(ii) の処理を終了した後，後処理として実現すれば十分である．

そこで，上述の (i),(ii) の処理をFPGAの配置処理と考え，評価指標として直接的に概略配線径路の混雑度を採用する．概略配線径路は，結線要求のある2つのLUTが割り当てられた異なる基本セル間を結線する場合，あるいは，IOブロックに割り当てられた外部入出力端子と基本セルとを結線する場合に途中に

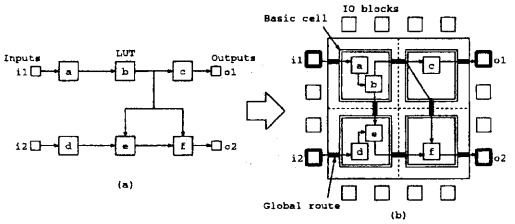

図 2: 配置・概略配線の例 (a) ネットリスト (b) レイアウト結果

どの基本セルを通ってこれらの間を結線するかによって表す（図1）．図1上の格子によって区切られる最小矩形を単位セルとしたとき（単位セルは1つの基本セルを含む），配線混雑度を，単位セルの外周となる各辺を通る概略配線数の最大値として定義する．概略配線径路は，正確に配線リソースに対応していないが，配線混雑度の評価には十分であると考える．

2.3 配置・概略配線問題

ネットをLUTおよび外部入出力端子間の結線要求とする．ネットリストをネットの集合とする[2]．上述の (i), (ii) の処理と共に概略配線径路を決定する，次のような配置・概略配線問題を定義する．

定義 1 本稿で扱う配置・概略配線問題とは，入力として，

(1) ネットリスト

(2) FPGA モデル（基本セルの列数および行数）

が与えられたとき，

(a) LUT によって構成される基本セル

(b) 基本セルの配置位置

(c) 基本セル間の概略配線径路

を配線混雑度が最小化するように出力することである．ただし，次の制約条件をとる．

- 使用する基本セル数は与えられた数を越えない．
- 各基本セルは高々4つの LUT を収容する．
- 外部入出力端子は IO ブロックに割り当てられる．

入力として与えられるネットリストおよび配置・概略配線結果の例を図2に示す．

3 配置・概略配線同時処理手法

3.1 基本アルゴリズム

アルゴリズムに必要となる以下の用語を定義する．

部分領域: 隣接した単位セルの集合によって構成される矩形領域．

[2]入力として，組合せ回路および順序回路をとる．順序回路は，LUT の出力をラッチするよう記述される．

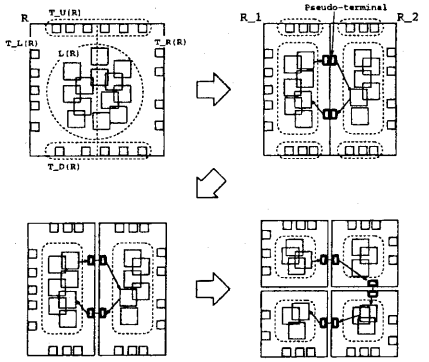

図 3: 部分領域および LUT 集合の分割処理

カットライン: 1つの部分領域を2つの部分領域に分割する水平あるいは垂直の線分．図1の格子に沿って引かれる．

仮想端子: カットラインによって分割された2つの部分領域にまたがるネットの結線要求を保つために，カットライン上に置かれる仮想的な端子．

$L(R)$: 部分領域 R内部に割り当てられた LUT の集合．

$T_L(R), T_R(R), T_U(R), T_D(R)$: 部分領域 Rの左辺，右辺，上辺，下辺に割り当てられた外部入出力端子あるいは仮想端子の集合．

提案手法の基本は，(a) レイアウト領域および LUT 集合の階層的2分割，(b) 分割された LUT 集合間の結線要求を保つための仮想端子による接続，にある[3]．処理の流れを以下に示す（図3参照）．

提案手法は，まず，レイアウト領域全体を1つの部分領域とみなし，部分領域の内部に LUT 集合，部分領域の4辺に外部入出力端子集合を割り当てる[4]．部分領域はカットラインに沿って分割される．部分領域 Rの分割に対応して，

(1) Rの分割される辺に割り当てられている端子集合の2分割処理

(2) Rの内部に割り当てられている LUT 集合の2分割処理

を実行する．この際，1つのネットで結線すべき LUT 同士が異なる部分領域に割り当てられた場合，結線要求を保つためにカットライン上に1ネットにつき1組の仮想端子を置く．仮想端子は，分割された各部分領域での外部入出力端子とみなすことができる．この時点では各仮想端子はカットライン上のどこに置くべきか決定されていない．部分領域が単位セルになるまで階層的に処理を進行することで，最終的に仮想端子は単位セルの外周に割り当てられる．1つのネットに属するこのような仮想端子の並びはネットの概略配線径路を表している．しかも，各単位

[3](a) は，min-cut 配置手法 [2] に見られる．(b) は，階層的概略配線手法 [4] に見られる．提案手法はこれら2つの処理を同時に処理するといえる．

[4]レイアウト領域の4辺への外部入出力端子の割当てでは，領域の左辺に入力端子を，領域の右辺に出力端子を割り当てるものとする．これは，通信処理用 FPGA が左辺から右辺への信号の流れを考慮して設計されているためである．設計者が端子位置を指定する場合は，レイアウト領域の4辺の指定位置に端子を配置する．

Step 1. 外部入出力端子集合をレイアウト領域の4辺に割り当てる．レイアウト領域全体を部分領域として待ち行列 $\mathcal{Q}$ に挿入する．

Step 2. $\mathcal{Q}$ から1つの部分領域 Rを取り出す．$\mathcal{Q}=\emptyset$ ならば処理を終了．

Step 3. Rをカットラインによって，2つの部分領域 R_1，R_2に分割する．カットラインは，Rを2等分するように Rの長辺を分割する．

Step 4. Rの分割に対応して，以下の処理を実行する．

4.1 分割された辺上に割り当てられている端子集合を2分割する．

4.2 領域内部に割り当てられているLUT集合を2分割する．

Step 5. R_1, R_2に割り当てられたLUTおよび端子間に結線要求がある場合，1ネットにつき1組の仮想端子をカットライン上に生成する．

Step 6. R_1および R_2が2つ以上の単位セルを含めば，$\mathcal{Q}$ に挿入する．Step 2へ．

図 4: 基本アルゴリズム

セルに高々4個のLUTを割り当て，かつ単位セル内に割り当てられたLUT集合によって1つの基本セルを構成すれば，概略配線径路が求まるのと同時に2.3節で述べた (i)，(ii) の処理を終了している．すなわち，階層処理の進行に伴って，LUTによる基本セルの構成，基本セルの配置処理，概略配線処理とが互いに影響を与えながら徐々に決定していく．このとき，配線混雑度は単位セルの外周に割り当てられた端子数の最大値に等しい．

以上の議論に基づいた配置・概略配線同時処理の基本アルゴリズムを図4に示す．基本アルゴリズムは，次の性質を持つ．

- 階層処理の初期段階では，ネットはLUT同士の結線要求を表す．互いに結線されるべきLUTが同じ部分領域に割り当てられるよう処理される．
- 階層処理が進むと，ネットは仮想端子とLUTの結線要求を表す．LUTは仮想端子で表される概略配線径路に従って，部分領域に割り当てられる．

図4のアルゴリズムにおいて，配線混雑度を小さくするには，Step 5で生成される仮想端子数をできるだけ小さく抑える必要がある．生成される仮想端子数は，Step 4の処理によって決まる．次節以降では，提案手法の核となるStep 4の処理を，垂直方向のカットラインを対象に配線混雑度の最小化の立場から議論する．

3.2 端子集合の分割

部分領域 Rの上辺および下辺に割り当てられている端子集合を2分割する（以下，仮想端子と外部入出力端子とを同一視し端子として扱う）．ここでは，上辺の端子集合の分割が終了したと仮定して，下辺の端子集合の分割について考える．

Rの下辺に割り当てられている端子集合 $T_D(R)$ を，

$$T_D(R_1) \cup T_D(R_2) = T_D(R) \ \wedge \ T_D(R_1) \cap T_D(R_2) = \emptyset$$

となるような2つの集合 $T_D(R_1)$ および $T_D(R_2)$ に分割し，各々カットラインの左側の部分領域 R_1の下辺および右側の部分領域 R_2の下辺に割り当てる．混雑度を小さくするため，カットラインの左右に対して等しい数の端子が割り当てられるよう分割する．すなわち，$|T_D(R_1)| = |T_D(R_2)|$ を目指す．さらに，カットライン上に生成される仮想端子数を小さくするため，カットラインの左側（あるいは右側）にすでに割当て済みの端子と推移的ファンイン，ファンアウト[5] となる端子が，分割された集合 $T_D(R_1)$（あるいは $T_D(R_2)$）に含まれるような分割を目指す．

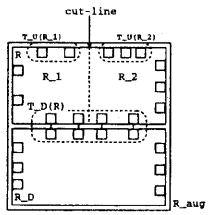

図 5: 拡大領域

$T_D(R)$ は，Rの下辺であるのと同時に，Rの下側に隣接している部分領域 R_Dにとって上辺であり，2つの領域のインターフェースをとっている．そこで，$T_D(R)$ の分割の際に Rと R_Dとを併合した領域 R_{aug}を処理の対象とする（図5）．

本節で用いる用語を定義する．

$dep(t_i, t_j)$: 2つの端子 $t_i, t_j \in T_D(R)$ が互いに推移的ファンイン，ファンアウトであるとき1，そうでないとき0をとる0-1変数．

$a_k(t_i)$: ある端子 $t_i \in T_D(R)$ が，分割後の部分領域 R_k $(k = 1,2)$ のいずれかに割り当てられたかを表す0-1変数で，部分領域 R_k $(k = 1,2)$ に属しているとき $a_k(t_i) = 1$，そうでないとき $a_k(t_i) = 0$ とする．

$nt_k(t_i)$: ある端子 $t_i \in T_D(R)$ を考える．R_{aug}の4辺に存在する端子のうち，(1) カットラインの左側（R_1側）にすでに割り当てられており，かつ (2) t_iと推移的ファンイン，ファンアウトとなっている端子の数を $nt_1(t_i)$ とする．同様に，カットラインの右側（R_2側）に関して $nt_2(t_i)$ とする．

$dep_k(t_i)$: ある端子 $t_i \in T_D(R)$ の部分領域 R_k $(k = 1,2)$ に対する依存度．$k = 1,2$ に対し，次のように定義する．

$$dep_k(t_i) = \frac{nt(t_i) + \sum_{t_j \in T_D(R)} dep(t_i, t_j) \cdot a_k(t_j)}{nt(t_i) + \sum_{t_j \in T_D(R)} dep(t_i, t_j)}$$

提案手法は，まず，各端子 $t_i \in T_D(R)$ について，依存度 $dep_k(t_i)$ を算出する．依存度 $dep_k(t_i)$ の値が大きいほど，端子 t_iは部分領域 R_kに割り当てられることが期待される．依存度が最大となる端子から，逐次，端子を部分領域 R_1あるいは R_2に割り当てる．図6に端子集合の2分割アルゴリズムを示す．

3.3 LUT集合の分割

部分領域 Rの内部に割り当てられているLUT集合 $L(R)$ を，

$$L(R_1) \cup L(R_2) = L(R) \ \wedge \ L(R_1) \cap L(R_2) = \emptyset$$

[5] 端子 t_1から入力された信号が，0個以上のLUTを経由してが端子 t_2に到達するとき，t_2を t_1の推移的ファンアウト，t_1を t_2の推移的ファンインと呼ぶ．

Step 1. $dep(t_i,t_j)$ $(t_i,t_j \in T_D(R))$ を算出する．$a_k(t_i) = 0$, $T_D(R_k) = \emptyset$ $(t_i \in T_D(R),\ k = 1,2)$ とする．

Step 2. 割当ての済んでいない各端子 $t_i \in T_D(R)$ の部分領域 R_k $(k = 1,2)$ に対する依存度 $dep_k(t_i)$ を算出する．

Step 3. 依存度が最大となっている端子を1つ選ぶ．そのような端子 t_i が部分領域 R_k に対して依存度が最大であれば，$a_k(t_i) = 1$, $T_D(R_k) = T_D(R_k) \cup \{t_i\}$ とする．$|T_D(R_k)| \geq |T_D(R)|/2$ になれば，まだ割当てが済んでいない端子をもう一方のチップに全て割り当て，終了．そうでなければ，Step 2へ．

図 6: 端子集合の2分割アルゴリズム

となるような2つの集合 $L(R_1)$ および $L(R_2)$ に分割し，各々カットラインの左側の部分領域 R_1 および右側の部分領域 R_2 に割り当てる．カットライン上に生成される仮想端子数を小さくするため，ネットがカットラインをまたがないような分割を目的とする．

R_1, R_2 内部の基本セル数を $N_B(R_1)$, $N_B(R_2)$ とする．簡単のため，以下の議論では $N_B(R_1) = N_B(R_2)$ を仮定する．各基本セルが高々4個のLUTを収容可能であることから，分割された各領域には，$N_L(R_1) = 4 \cdot N_B(R_1)$ 個および $N_L(R_2) = 4 \cdot N_B(R_2)$ 個のLUTを割り当てることができる．ところが初期の階層処理で1つ部分領域に割当て可能なこれらのLUTを使い切ると，その領域での混雑度を増加させるため望ましくない．ここでは，各部分領域に割り当てられるLUT数を次式のように制約する．

$$|L(R_k)| \leq M_k \ (k = 1,2)$$

ただし，

$$M_k = \frac{|L(R)|}{2} + a \cdot \left(N_L(R_k) - \frac{|L(R)|}{2} \right)$$

ここで $a = 1/\lg(N_B(R_k)+1)$ とする[6]．階層処理の初期では a の値が小さく，R_k がただ1つの単位セルとなるとき $a = 1$ となる．上式は，$a = 0$ のとき $L(R)$ を完全に2等分し，$a = 1$ のとき部分領域内部に含まれるすべての基本セルを使いきるような割当てを許容する．このような制約を付加することにより，初期の階層処理での混雑度の増加を防ぐことが期待できる．LUT数に関するこの制約をサイズ制約と呼ぶ．

本節では最大フローによる最小カット探索を用いて，LUT集合の分割を実現する．最大フローによるカットの算出は，サイズ制約を満足するとは限らないが，厳密に最小カットを発見可能な数少ない手法の1つである．配線混雑度を小さく抑えるには，このような手法の適用が不可欠である．

本節で用いる用語を定義する．

$G^R = (V,E)$: 部分領域 R に含まれるネットリストから構築される容量付きグラフ．G の各節点は，R の内部のLUT，R の外周の端子，それらの結線要求を表すネットのうちいずれかに対応する[7]．これらに対応する G の節点集合をそれぞれ V_L, V_T, V_N と書く．

G の枝 $e \in E$ は，各ネットに対して図7のようにとる [17]．すなわち，各ネット n および n によって結線されるLUTおよび端子の集合 L_n に対して，

[6] $N_B(R_1) \neq N_B(R_2)$ のとき，上式において $|L(R)|/2$ の部分を，$|L(R)|$ を $N_B(R_1)$ と $N_B(R_2)$ の比に応じて分割した数に置き換える．

[7] すぐ後で述べるように，G 内にネットを表す節点は各ネットに関して2つずつ存在する．

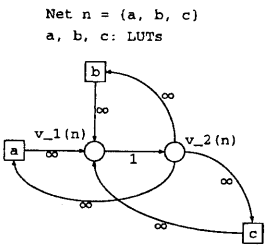

図 7: ネットに対応する容量付きグラフ

- ネットを表す2節点 $v_1(n)$ および $v_2(n)$ を用意する．
- 各 $v \in L_n$ と $v_1(n)$ とを容量 ∞ の枝 $(v, v_1(n))$ で接続．
- 各 $v \in L_n$ と $v_2(n)$ とを容量 ∞ の枝 $(v_2(n), v)$ で接続．
- 用意された2節点 $v_1(n)$ および $v_2(n)$ は容量1の枝 $(v_1(n), v_2(n))$ で接続する．

このような枝を考えることで，ネットリストの分割にネットワークフローの考えを適用することができる．

節点集合 V_T は，カットラインの左側および右側に割り当てられた端子に対応する節点集合 $V_{T_1} = T_L(R) \cup T_U(R_1) \cup T_D(R_1)$ および $V_{T_2} = T_R(R) \cup T_U(R_2) \cup T_D(R_2)$ に分けられる．

$G^R_{st} = (V', E')$: G に対し，ソース節点 s，シンク節点 t を考え，s と $v \in V_{T_1}$ とを容量 ∞ の枝 (s,v)，t と $u \in V_{T_2}$ とを容量 ∞ の枝 (u,t) で結んだグラフを G^R_{st} とする．枝 $e \in E'$ の容量を $c(e)$ と書く．

$(X, \overline{X})$: $s \in X$, $t \in \overline{X}$ となるような G^R_{st} の節点集合の分割．カットと呼ぶ．

$E_f(X,\overline{X})$, $E_b(X,\overline{X})$: カット $(X,\overline{X})$ と交差する枝のうち，その向きが X から $\overline{X}$ の方向となっている枝集合を $E_f(X,\overline{X})$，逆方向となっている枝集合を $E_b(X,\overline{X})$ と書く．

$sz(X,\overline{X})$: カット $(X,\overline{X})$ のカットサイズ．

$$sz(X,\overline{X}) = \sum_{e \in E_f(X,\overline{X})} c(e)$$

$V_L(X), V_L(\overline{X})$: カット $(X,\overline{X})$ が与えられたとき，各々 X あるいは $\overline{X}$ に含まれるLUTの集合．すなわち，$V_L(X) = V_L \cap X$, $V_L(\overline{X}) = V_L \cap \overline{X}$.

$V_s(X), V_t(\overline{X})$: カット $(X,\overline{X})$ が与えられたとき，各々 X, $\overline{X}$ に含まれる節点で，ソース節点 s と接続しているものを $V_s(X)$，シンク節点 t と接続しているものを $V_t(\overline{X})$ と書く．

容量付きグラフ G^R_{st} のカット $(X,\overline{X})$ を考える．LUT集合 $L(R)$ を，カットによって $V_L(X)$ および $V_L(\overline{X})$ の2つの集合に分割すれば，ネットのグラフ表現から，カット $(X,\overline{X})$ のカットサイズは，丁度，部分領域 R のカットラインを横切るネット数，すなわち生成される仮想端子の組数に等しい．そこで，G^R_{st} のカット $(X,\overline{X})$ を，サイズ制約を満たすよう，かつカットサイズが小さくなるように探索することを考える．最大フロー最小カットに基づく次のアプローチをとる．

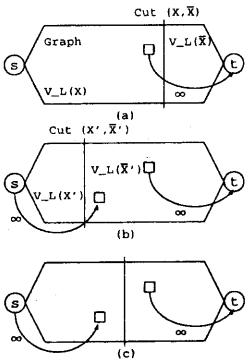

図 8: 最小カットの繰り返し探索 (a) 1 回目の反復, (b) 2 回目の反復, (c) 3 回目の反復（サイズ制約の満足）

(1) G^R_{st}において s から t に至る最大フローを求め，最小カット $(X, \overline{X})$ を求める．

(2) カットがサイズ制約を満たしていれば終了．

(3) 満たしていなければ，$V_L(X) > M_1$のとき t と $v \in V_L(X)$ を接続し，$V_L(\overline{X}) > M_2$のとき s と $v \in V_L(\overline{X})$ を接続する．(1) へ．

図 8に動作例を示す．まず，1回目の反復で，最小カット $(X, \overline{X})$ が探索される．$V_L(X)$ が大きくサイズ制約を満たさないとき，X 内の 1 つの節点とシンク節点 t とを容量 ∞ の枝で接続する．枝を付加することで，カット $(X, \overline{X})$ は，もはや最小カットでなくなる．そのため 2 回目の反復では，1 回目とは異なる位置に最小カット $(X', \overline{X'})$ が算出される．$V_L(X')$ が小さくサイズ制約を満たさないとき，1 回目と同様に，X'内の 1 つの節点とソース節点 s とを容量 ∞ の枝で接続する．枝を付加することで，カット $(X', \overline{X'})$ は，もはや最小カットでなくなる．3 回目の反復で，最終的にサイズ制約を満たすカットを発見することができる．このアプローチは，最小カットを繰り返し探索することで，サイズ制約を満足しかつカットサイズを小さく抑えたカットを発見することが期待される．

図 9に LUT 集合の 2 分割アルゴリズムを示す．アルゴリズム中，Step 0，Step 2 は自明である．Step 1 は augmenting path 法 [12] を用いることで実現される．

Step 3.1（Step 3.2 の場合も同様）では，信号ネットの方向と LUT の入出力端子位置とを合致させるため，次のように t と接続される LUT $v \in V_L(X) - V_s(X)$ を選択する．すなわち節点 $v \in V_L(X) - V_s(X)$ として，

(a) カットと隣接し[8]，かつ

(b) その LUT の出力信号を入力とする LUT の数が X内に最も少ない LUT

を選択し，vと tとを容量 ∞ の枝 (v, t) で接続する．節点の選択例を図 10に示す．図は，G^R_{st}そのものでなく，LUT を矩形，その間の信号の流れを矢印で示している．LUT が図中のカットで

[8] カット $(X, \overline{X})$ に対し，LUT $v \in X$の出力信号を入力とする LUT $u \in \overline{X}$ が存在するとき，u, vは，カット $(X, \overline{X})$ に隣接するという．

Step 0. 部分領域 Rの中に含まれるネットリストから $G^R_{st} = (V', E')$ を構築する．

Step 1. G^R_{st}の最大フローを求めることで，$sz(X, \overline{X})$ が最小かつ $|X|$ が最小のカット $(X, \overline{X})$ を算出する．

Step 2. $|V_L(X)| \leq M_1 \wedge |V_L(\overline{X})| \leq M_2$ならば，$L(R_1) \leftarrow V_L(X)$, $L(R_2) \leftarrow V_L(\overline{X})$ として，終了．

Step 3. 次のどちらかの処理を実行する．

3.1 $|V_L(X)| > M_1$ならば，シンク節点 t と各節点 $u \in \overline{X}$ とを容量 ∞ の枝 (u, t) で接続する．さらに，t と 1 つの節点 $v \in V_L(X) - V_s(X)$ とを容量 ∞ の枝 (v, t) で接続する．Step 1 へ．

3.2 $|V_L(\overline{X})| > M_2$ならば，ソース節点 s と各節点 $v \in X$ とを容量 ∞ の枝 (s, v) で接続する．さらに，s と 1 つの節点 $u \in V_L(\overline{X}) - V_t(\overline{X})$ とを容量 ∞ の枝 (s, u) で接続する．Step 1 へ．

図 9: 最大フローに基づく LUT 集合の分割アルゴリズム

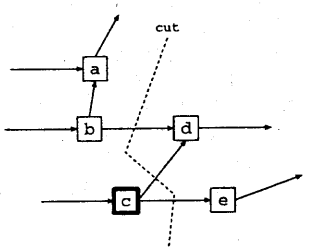

図 10: t と接続する LUT の選択 ($a, b, c \in V_L(X)$, $d, e \in V_L(\overline{X})$ のとき，LUT c が t と接続される)

2 つの集合 $V_L(X)$（カットの左側）と $V_L(\overline{X})$（カットの右側）に分割されたとき，t と $V_L(X)$ 内の 1 つの LUT とを接続することを考える．LUT b, c は共にカットに隣接している．bの出力信号は X内の LUT a に入力されているが，cの出力信号は X 内の LUT に入力されていない．この場合，cが t と接続される LUT として選択される．

このように Step 3 の処理を実現することで，信号ネットは $V_L(X)$ の方から $V_L(\overline{X})$ の方へ向かう流れとなり，LUT の入出力端子位置の方向と合致する．

3.4 計算複雑度

計算複雑度を見積もる．提案手法は部分領域の階層的分割に基づくため，各部分領域はレベルを持つ．レイアウト領域全体をレベル 1 とする．レベル lの部分領域によって生成される 2 つの部分領域のレベルを $(l+1)$ とする．与えられる基本セルの数を $L_1 \times L_2$ $(L_1 \geq L_2)$ とすれば，単位セルのレベルは $O(\lg L_1)$ となる．N_Lを与えられた LUT および外部入出力端子の合計数，N_P を最終的に生成された仮想端子数とし，$N = N_L + N_P$とする．なお，ネット数を N_Nとするとき，N_Pは高々$O(N_N \cdot (L_1 \times L_2))$ で抑えられる．

レベル lのある部分領域 Rの分割処理を考える．この際，最も時間を要するのは LUT 集合の分割である．図 9において，G^R_{st} の節点数を $N_V = |V'|$ とすれば，G^R_{st}の構成法から最大フローの流量は高々N_Vである．図 9の Step 3 の処理は $O(N_V)$ の時間を要するため，全体で $O(N_V^2)$ の時間を要する．従って，レベル

lの全部分領域に関しては，全体で$O(N^2)$の時間を要する．

以上の議論から，$O(\lg L_1)$レベル全体において，提案手法の時間複雑度は$O(N^2 \cdot \lg L_1)$となる．空間複雑度は$O(N)$である．

4 計算機実験結果

提案手法をSUN Sparc Station 2（28.5MIPS）上にC言語を用いて実装した．実際の通信処理用FPGAチップPROTEUS[15]に対し提案手法を適用し評価実験を行った．表1に各実験回路の外部入出力数（#input，#output），LUT数（#LUT）およびラッチ数（#FF）を示す．回路は，mpa_ty1, 2を除き，動作記述（SFLファイル）からPARTHENON[7][9]により論理合成を行い，PROTEUS専用CAD（PROTEUS-CAD）[14]に含まれるprocoverによりテクノロジーマッピングしたものである．

比較のため次の2つの実験を試みた．

(1) **提案手法:** 実際のPROTEUSチップを対象として，表1の回路に提案手法を適用する．外部入出力端子は，mpa_ty2を除いて，左辺のIOブロックに入力端子，右辺のIOブロックに出力端子を割り当てる．mpa_ty2は，予め設計者によって決められたIOブロック位置に外部入出力端子を割り当てる．

(2) **profix:** まず，表1の回路にPROTEUS-CADの配置ツールprofixを適用し，配置処理を実行する[10]．PROTEUS-CADは概略配線ツールを持たない．ここでは，profixの配置結果に対し階層的概略配線手法[4]を適用し，概略配線処理を実行した．外部入出力端子は，mpa_ty2を除いて，どのIOブロックに割り当ててもよいとする．mpa_ty2は，予め設計者によって決められたIOブロック位置に外部入出力端子を割り当てる．

表2，3に実験結果を示す．profixの処理時間は，profixによる配置の処理時間と階層的概略配線の処理時間との和の形で示している．表中，仮想配線長（est. wire length）は次のように算出した．(1)提案手法，profix共に配置結果のみを取り出す．(2)各ネットを出力端子，入力端子間を結ぶ2端子間ネットに分解する．(3)分解された全ネットの配線長をマンハッタン距離（単位セルの1辺を単位長とした）で表し，その和を算出する．仮想配線長は，profixの第一の目的関数である．表から，提案手法はprofixに比較して，仮想配線長を平均30%増加させているが配線混雑度を平均37%減少していることが分かる．配線混雑度の減少には，配置と概略配線とを同時に処理していくことが不可欠であることが結論づけられる．しかも，提案手法の処理時間は数秒程度である．提案手法を用いることにより，FPGAを用いてハードウェア設計を繰り返すような場合でも，短時間で再設計を実現できる．

さらに，提案手法の有効性を確かめるため，提案手法，profixによって得られた結果に対し詳細配線を実行した．配線処理はPROTEUS-CADの配線ツールprorouteを使用した．prorouteは，配線処理を概略配線，詳細配線に分割せず，配置結果から直接，詳細配線結果を生成する．そのため，提案手法，profix共に配置結果のみを取り出して，prorouteによる配線を試みた．表4，5に実験結果を示す．表は，未配線数（#unroutes）および回路の最大遅延の見積り値（max. d）[11]を示す．表から，profixの配線結果がいくつかの回路に対して未配線を生じているのに対し，提案手法の配線結果ではほぼ全回路において100%配線を実現している[12]．しかも，最大遅延はprofixによるものとほぼ同等である．提案手法が最大遅延を小さく抑えることができたのは，結線要求の高いLUT同士が比較的同じ部分領域に割り当てられたためと考えられる．

以上の結果から，提案手法は配線混雑度を小さく抑えることで，より100%配線を容易にする設計を実現することが実験的に確かめられた．

表 1: 実験回路

circuit	#inputs	#outputs	#LUTs	#FFs
bip24	11	25	125	72
bip24ed	36	1	57	24
bip8	10	9	43	24
cnt_3	3	3	8	3
cnt_4	3	4	12	4
cnt_5	3	5	16	5
cnt_6	3	6	20	6
cnt_8	3	8	28	8
cnt_9	3	9	30	9
mpa_ty1	20	22	414	206
mpa_ty2	20	22	414	206
osync	11	8	114	18
scr	11	8	50	8

5 むすび

通信処理用FPGAを対象とした配置・概略配線同時処理手法を提案した．提案手法は，配置処理と同時に概略配線処理を実行することで，配線混雑度の小さく抑えたレイアウトを生成する．計算機実験により手法の有効性を確認した．

提案手法は，現段階で従来手法とほぼ同程度の遅延時間を得ているが，さらに動作周波数を向上させるため，今後，提案手法に配線遅延に関する制約を付加することが必要である．

謝辞

PROTEUS-CADをご提供していただきました太田直久氏，宮崎敏明氏，筒井章博氏をはじめとするNTT光ネットワークシステム研究所の方々に感謝致します．本研究の一部は，文部省科学研究補助金（特別研究員奨励費）の補助を受けた．

参考文献

[1] M. Burstein, S. J. Hong, and R. Pelavin, "Hierarchical VLSI layout: Simultaneous placement and wiring of gate arrays," in *Proc. VLSI'83*, pp. 45–60, 1983.

[2] M. A. Breuer, "Min-cut placement," *J. Design Automation and Fault Tolerant Computing*, vol. 1, no. 4, pp. 343–362, 1977

[3] W.-M. Dai and E. S. Kuh, "Simultaneous floor planning and global routing for hierarchical building-block layout," *IEEE Trans. Comput.-Aided Des. Integrated Circuits & Sys.*, vol. CAD-6, no. 5, pp. 828–837, 1987.

[9] PARHTENONをご提供していただきましたNTTおよびNTTデータ通信の方々に感謝します．

[10] profixのパラメータは，標準スクリプト（ProMakefile）のものを用いた．

[11] ここでは，prorouteによって見積もられた値を示した．

[12] 提案手法においてmpa_ty1, 2では，各々2，1ネットの未配線を生じたが，予め未配線となったネットを配線してから，残りのネットを配線したところ，100%配線を実現できた．

表 2: 配線混雑度に関する実験結果 (ours)

circuit	channel density	est. wire length	CPU time[s]
bip24	7	2585	3.23
bip24ed	9	1128	2.26
bip8	7	421	0.48
cnt_3	3	114	0.09
cnt_4	5	174	0.14
cnt_5	5	184	0.19
cnt_6	4	366	0.22
cnt_8	4	496	0.47
cnt_9	4	526	0.45
mpa_ty1	11	4266	11.62
mpa_ty2	13	4544	11.27
osync	9	2702	5.17
scr	7	974	1.57
total	88	18480	37.16

表 3: 配線混雑度に関する実験結果 (profix)

circuit	channel density	est. wire length	CPU time[s]
bip24	17	1710	202.07+ 4.70
bip24ed	11	726	237.19+ 1.53
bip8	10	454	205.64+ 1.01
cnt_3	5	51	228.88+ 0.23
cnt_4	4	82	222.99+ 0.22
cnt_5	5	149	214.91+ 0.26
cnt_6	4	168	207.90+ 0.28
cnt_8	6	176	203.22+ 0.33
cnt_9	4	233	207.94+ 0.37
mpa_ty1	20	3506	183.71+10.96
mpa_ty2	29	4001	161.17+19.14
osync	16	1316	178.63+ 3.33
scr	10	363	202.43+ 0.64
total	141	12935	2656.68+43.00

表 4: 配線達成率に関する実験結果 (ours)

circuit	#unroutes	CPU time [s]	max. d [ns]
bip24	0	399	38.5
bip24ed	0	275	40.0
bip8	0	96	20.0
cnt_3	0	39	10.0
cnt_4	0	42	17.0
cnt_5	0	54	14.5
cnt_6	0	61	30.5
cnt_8	0	88	47.0
cnt_9	0	74	26.5
mpa_ty1	2(0)†	912	59.0
mpa_ty2	1(0)†	963	56.0
osync	0	273	68.5
scr	0	168	48.0
total	3(0)	3444	475.5

†はじめに配線処理を実行した際に未配線が生じた．そのため，配線順序を変更し，未配線となったネットを先に配線し，その後残りのネットを配線した結果，未配線を 0 にすることができた．

表 5: 配線達成率に関する実験結果 (profix)

circuit	#unroutes	CPU time [s]	max. d [ns]
bip24	7	727	-
bip24ed	3	270	-
bip8	0	98	24.5
cnt_3	0	41	17.5
cnt_4	0	45	21.5
cnt_5	0	63	19.0
cnt_6	1	83	-
cnt_8	1	67	-
cnt_9	0	69	30.5
mpa_ty1	20	896	-
mpa_ty2	25	891	-
osync	21	502	-
scr	2	135	-
total	80	3887	(113.0)

[4] U. P. Lauther, "Top down hierarchical global routing for channelless gate arrays based on linear assignment," in *Proc. VLSI'87*, pp. 109–120, 1987.

[5] S. K. Nag and R. A. Rutenbar, "Performance-driven simultaneous place and route for row-based FPGAs," in *Proc. 31st DA Conf.*, pp. 301–307, 1994.

[6] S. K. Nag and R. A. Rutenbar, "Performance-driven simultaneous place and route for island-style FPGAs," in *Proc. ICCAD-95*, pp. 332–338, 1995.

[7] Y. Nakamura, K. Oguri, A. Nagoya, M. Yukishita, anr R. Nomura, "High-level synthesis design at NTT system labs," IEICE Trans. Inf. & Syst., vol. E76-D, no. 9, 1993.

[8] M. Ohmura, S. Wakabayashi, Y. Toyohara, J. Miyao, and N. Yoshida, "Hierarchical floorplanning and detailed global routing with routing-based partitioning," in *Proc. ISCAS'90*, pp. 1640–1643, 1990.

[9] N. Ohta, H. Nakada, K. Yamada, A. Tsutsui, and T. Miyazaki, "PROTEUS: Programmable hardware for telecommunication systems," in *Proc. ICCD'94*, pp. 178–183, 1994.

[10] 佐藤政生, 戸川望, "FPGA 用のレイアウト手法," 情報処理学会誌, vol. 35, no. 6, pp. 535–540, 1994.

[11] P. R. Suaris and G. Kedem, "A quadrisection-based combined place and route scheme for standard cells," *IEEE Trans. Comput.-Aided Des. Integrated Circuits & Sys.*, vol. 8, no. 3, pp. 234–244, 1989.

[12] R. E. Tarjan, *Data Structures and Network Algorithms*, Society for Industrial and Applied Mathematics, 1983.

[13] N. Togawa, M. Sato, and T. Ohtsuki, "Maple: A simultaneous technology mapping, placement, and global routing algorithm for field-programmable gate arrays," in *Proc. ICCAD-94*, pp. 156–163, 1994.

[14] A. Tsutsui and T. Miyazaki, "An efficient design environment and algorithms for transport processing FPGA," in *Proc. VLSI'95*, pp. 791–798, 1995.

[15] A. Tsutsui, T. Miyazaki, K. Yamada, and N. Ohta, "Special purpose FPGA for high-speed digital telecommunication system," in *Proc. ICCD'95*, pp. 486–491, 1995.

[16] Xilinx, *The Programmable Logic Data Book*, 1993.

[17] H. Yang and D. F. Wong, "Efficient network flow based min-cut balanced partitioning," in *Proc. ICCAD-94*, pp. 50–55, 1994.