設 計 自 動 化 81－6
(1996. 10. 18)

パス長制約を考慮した通信処理用 FPGA 向け配置・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

通信処理用 FPGA を対象としたレイアウト合成では，配線混雑度を小さく抑えると共に高速動作を可能とした回路設計が要求される．本稿では，通信処理用 FPGA を対象に配線混雑度最小化を目的として提案された配置・概略配線同時処理手法を拡張し，タイミングがクリティカルな信号パスの長さ（パス長）の最大値を制約として与え，パス長を最大値以内に抑えることを可能とした手法を提案する．提案手法は階層的 2 分割に基づく．各 2 分割処理は，(0) パス長制約の評価，(1) 端子集合の 2 分割，(2) LUT 集合の 2 分割，の 3 段階より構成される．(0) により，パス長制約がより厳しいパスを探索し，(1)，(2) でそのようなパスのパス長が優先的に短くなる処理を実行することでパス長制約の満足を目指す．提案手法をいくつかの通信処理用回路に適用し評価実験した結果を報告する．

キーワード　通信処理用 FPGA，レイアウト設計，配置，概略配線，パス長制約

A Performance-Driven Simultaneous Placement and 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

In layout design of transport-processing FPGAs, it is required that not only routing congestion is kept small but also circuits implemented on them operate with higher operation frequency. This paper extends the proposed simultaneous placement and global routing algorithm for transport-processing FPGAs whose objective is to minimize routing congestion and proposes a new algorithm in which the length of each critical signal path (path length) is limited within a specified upper bound imposed on it (path length constraint). The algorithm is based on hierarchical bipartitioning of layout regions and LUT (LookUp Table) sets to be placed. Each bipartitioning procedure consists of three phases: (0) estimation of path lengths, (1) bipartitioning of a set of terminals, and (2) bipartitioning of a set of LUTs. After searching the paths with tighter path length constraints by estimating path lengths in (0), (1) and (2) are executed so that their path lengths are reduced with higher priority and thus path length constraints are not violated. The experimental results demonstrate the efficiency and effectiveness of the algorithm.

Key Words *transport-processing FPGA, layout design, placement, global routing, path length constraint*

1 まえがき

特定用途向けのFPGA（Field-Programmable Gate Arrays）として，通信処理専用FPGAが提案されている[4],[8],[9][1]．通信処理用FPGAは，文献[10]に代表される一般的なFPGAに比較して，論理機能を実現する論理単位の粒度が細かく，より柔軟な回路を実現することができる．しかも，通信処理は信号の流れの方向性が強いことから，セルの入出力端子位置が揃っている．このような通信処理用FPGAの配置・概略配線設計を考えた際，まず次の点が満足される必要がある．

(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．

加えて，連続かつ大量に入力される通信情報をリアルタイムに処理するために，次の点が満足される必要がある．

(2) 通信処理用FPGA上に実現される回路は，その動作周波数ができるだけ高い方が望ましい．

いくつかの通信処理用FPGAのレイアウト合成手法が提案されている[8],[6]．文献[8]では，min-cut配置手法およびクリティカルパス遅延の最小化を目的とした迷路法による配線手法が提案されている．ところが，この手法は配線混雑度の小さいレイアウトを生成することが難しく，上述の(1)の要求を満足しない．文献[6]では，配置・概略配線同時処理手法が提案されている．配線混雑度を直接評価するため，配線混雑度が小さく，その結果100%配線を容易に実現可能なレイアウトを生成できる．反面，配線遅延に関しては陽に考慮されておらず，このままでは上述の(2)の要求を満足していない．

一般の性能指向FPGA合成手法として，文献[1],[5]が提案されている[2]．文献[1]では，テクノロジーマッピング・配置を同時実行し配線遅延を最小化する．ところが，配線遅延の見積りに配置処理の結果のみを考えるため，正確な配線情報による遅延を得ることは難しい．文献[5]では，性能指向テクノロジーマッピング・配置・概略配線同時処理手法が提案されている．この手法は，文献[2]にあるFPGAに対し最適化されているため，通信処理用FPGAに適用するのは難しい．

以上のような背景から，本稿では，提案された通信処理用FPGAを対象とした配置・概略配線同時処理手法[6]を拡張し，タイミングがクリティカルな信号パスの長さ（パス長）の最大値を制約として与え，パス長を最大値以内に抑えることを可能とした手法を提案する．

2 準備

2.1 FPGAモデル

文献[9]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I/Oブロックが整列する．

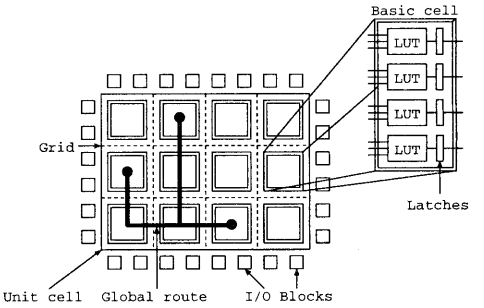


図 1: FPGAモデル．

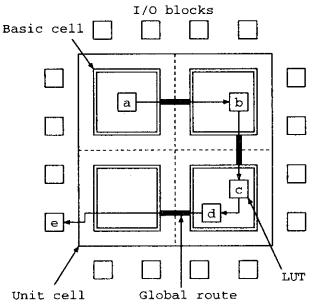


図 2: パス長．

2.2 パス長制約

配置すべきLUT集合および外部入出力端子集合の結線要求をネットと呼ぶ．ネットの集合をネットリストと呼ぶ．ネットの概略径路は，途中に通過する基本セルの並びによって表される[6]．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定義する．

いま，ネットリストに対し配置・概略配線が与えられたとする．信号パスとは，1つのLUTあるいは外部入力によって与えられる信号が，いくつかのLUTを経由し，別のLUTあるいは外部出力に至るパスをいう．信号パス p のパス長 $l(p)$ を p が通過した単位セルの辺の数によって定義する（外周の辺数を除く）．例えば，図2は 2×2 の基本セルから成るFPGAを表し，LUT a によって与えられる信号が，LUT b, c, d を経由しLUT e に至る信号パス p に対し，パス長 $l(p) = 3$ を与える．

パス長制約とは，配置・概略配線を実行する前に，信号パス p のパス長の上限値 $l_{max}(p)$ を与え，パス長 $l(p)$ を

$$l(p) \leq l_{max}(p)$$

となるように配置・概略配線することをいう．パス長制約は，タイミングがクリティカルとなる複数の信号パスに対して与えられる．

2.3 配置・概略配線問題

次のような配置・概略配線問題を定義する．

定義 1 配置・概略配線問題とは，入力として，

[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[4]．

[2] 文献[6]にあるように，通信処理用FPGAを対象とした配置・概略配線問題は，広義のテクノロジーマッピング・レイアウト問題と考えられる．ここでは，FPGA合成手法の中でテクノロジーマッピングとレイアウト合成を同時に実行可能な手法を取り上げた．

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 3: 基本アルゴリズム．

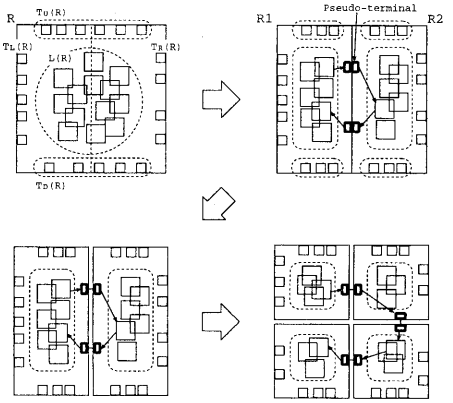


図 4: 部分領域および LUT 集合の分割処理．

(1) ネットリスト

(2) FPGA モデル（基本セルの列数および行数）

(3) パス長制約の集合

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 I/O ブロックに割り当てられる．
- パス長制約を満足する．

3 パス長制約を考慮した配置・概略配線同時処理手法

3.1 基本アルゴリズム

提案手法は，文献 [6] による配置・概略配線手法の拡張であり，その基本アルゴリズムを継承する．基本アルゴリズムに必要となる以下の用語を定義し，基本アルゴリズムを図 3 に示す．

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．

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．

$w(R), h(R)$: 部分領域 Rの幅および高さ．単位セルの1辺の長さを単位とする．

基本アルゴリズムの特徴は，(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（図4参照）．階層的2分割を部分領域が1つの単位セルを含むまで繰り返し，かつ各単位セルに高々4 個の LUT を割り当てることにより，最終的に4個の LUT によって形成される基本ブロック，基本ブロックの配置，仮想端子の並びによる概略配線径路を同時に得ることができる．

基本アルゴリズムにおける各2分割処理は，(1) 端子集合の2分割，(2) LUT 集合の2分割の2段階から構成される．しかしながら，このままではパス長制約の満足は望めない．そこで，パス長制約の導入にあたり，まず，各階層処理のはじめに，(0) パス長制約が課されたパスの中で，現地点の階層でパス長制約がより厳しいパスを探索し，続いて，パス長制約がより厳しいパスが優先的に1つの部分領域に収容されるように (1)，(2) の処理を拡張する．

以下，処理 (0)，拡張された処理 (1)，(2) について，垂直方向のカットラインを対象に，各々3.2 節，3.3 節および 3.4 節で提案する（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）．

3.2 パス長制約の評価

階層処理において，これまでのレイアウト結果に基づき制約が課されたパス（制約パスと呼ぶ）のパス長の下限値を算出する．この下限値と制約として与えられる上限値との差をとることにより，そのパスの制約の厳しさを評価する．

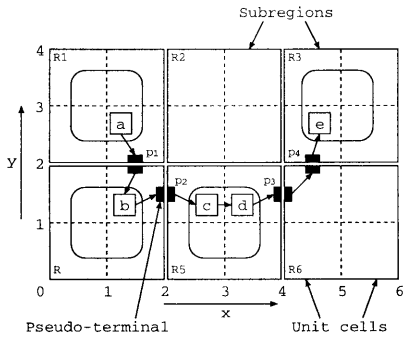
パス p のパス長の下限値を次のように算出する．パス長は，パス上に生成される仮想端子組の数に等しい．いま，パス p が，これまでの階層処理によって k個の仮想端子の組から構成されているとする．パス p を構成する仮想端子の組のうち，縦方向のカットライン上にあるものの集合 $PT_v(p)$ を信号の流れに沿って順に，

$$PT_v(p) = \{p_1^v, p_2^v, \ldots, p_m^v\}$$

とする．同様に，横方向のカットライン上にある仮想端子の組の集合を

$$PT_h(p) = \{p_1^h, p_2^h, \ldots, p_n^h\}$$

とする．但し，$k = m + n$ である．縦方向（あるいは横方向）のカットライン上にある2組の仮想端子に対し，信号が流れる方向が，共に左から右（あるいは上から下）またはその逆で一致している場合，その2組の仮想端子は方向が一致しているという．レイアウトモデル上のグリッドに対し，図5のように座標軸をとったとき，各 $p_i^v \in PT_v(p)$ の x 座標を $x(p_i^v)$，各 $p_i^h \in PT_h(p)$ の y座標を $y(p_i^v)$ と書く．このとき，



Subregions: $\{R_1, R_2, R_3, R_4, R_5, R_6\}$
Pairs of pseudo-terminals: $\{p_1, p_2, p_3, p_4\}$
Path $p = a \rightarrow b \rightarrow c \rightarrow d \rightarrow e$

図 5: パス長の下限値の算出.

$$l_{low}(p) = k + \sum_{2 \le i \le m} [|x(p^v_{i-1}) - x(p^v_i)| - d(p^v_{i-1}, p^v_i)] + \sum_{2 \le j \le n} [|y(p^h_{j-1}) - y(p^h_j)| - d(p^h_{j-1}, p^h_j)]$$

但し,

$$d(p_1, p_2) = \begin{cases} 1 & \left(\begin{array}{c} 2\text{組の仮想端子}\ p_1\text{および}\ p_2 \\ \text{の方向が一致している} \end{array} \right) \\ 0 & (\text{一致していない}) \end{cases}$$

とする. $l_{low}(p) \le l_{max}(p)$ が必ず成り立ち, l_{low}は, p のパス長の下限値を与えている.

例えば, 図5の例を考える. パス $p = a \rightarrow b \rightarrow c \rightarrow d \rightarrow e$ は, これまでの階層処理で $k = 4$ 個の仮想端子組を持ち, $PT_v(p) = \{p_2, p_3\}$, $PT_h(p) = \{p_1, p_4\}$ である. このとき下限値 $l_{low}(p)$ は,

$$\begin{aligned} l_{low}(p) &= k + [|x(p_2) - x(p_3)| - 1] + [|y(p_1) - y(p_4)| - 0] \\ &= 4 + (4 - 2 - 1) + (2 - 2 - 0) \\ &= 5 \end{aligned}$$

となる. パス p は, すでにある4個の仮想端子組および部分領域 R_5の通過のため, さらに少なくとも1つの仮想端子組を必要とする. そのため, 現時点のパス長の下限値は5となる.

パス長の下限値と制約値との差をスラック $slack(p)$ とする.

$$slack(p) = l_{max}(p) - l_{low}(p).$$

スラックが小さいほど, そのパスのパス長制約は厳しく, よりタイミングがクリティカルなパスであるといえる. 制約パスのスラックの算出は, 端子集合の2分割処理およびLUT集合の2分割処理の前に実行する. パス長制約の課されていないパスのスラックは便宜上 ∞ とする.

3.3 端子集合の2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. 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.

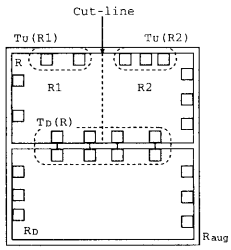


図 6: 拡大領域.

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. パス長制約が厳しいパスを1つの部分領域に収容することが望ましい. そのため, パス長制約が厳しいパスほど, その推移的ファンイン, ファンアウト[3]となる外部入出力端子同士が同じ部分領域に割り当てられるよう処理する.

$T_D(R)$ 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6) [6].

本節で用いる用語を定義する.

$T^p_L(R_{aug}), T^p_R(R_{aug})$: $T_D(R)$ の分割前に, すでにカットラインに対して R_{aug}の左側 (右側) に割り当てられている端子集合. 図6では, $T^p_L(R_{aug}) = T_U(R_1) \cup T_L(R_{aug})$, $T^p_R(R_{aug}) = T_U(R_2) \cup T_R(R_{aug})$ である.

$dep(t_i, t_j)$: 2つの端子 $t_i, t_j \in T_D(R) \cup T^p_L(R_{aug}) \cup T^p_R(R_{aug})$ 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 $t_i \in T_D(R) \cup T^p_L(R_{aug}) \cup T^p_R(R_{aug})$ が, 部分領域 R_k $(k = 1, 2)$ のいずれかに属しているかを表す0-1変数. $t_i \in T^p_L(R_{aug})$ ならば $a_1(t_i) = 1 \ \wedge \ a_2(t_i) = 0$, $t_i \in T^p_R(R_{aug})$ ならば $a_1(t_i) = 0 \ \wedge \ a_2(t_i) = 1$ とする. $t_i \in T_D(R)$ ならば, t_iが部分領域 R_k $(k = 1, 2)$ に割り当てられたとき $a_k(t_i) = 1$, そうでないとき $a_k(t_i) = 0$ とする.

ここでは, $dep(t_i, t_j) = 1$ なる端子 t_i, t_jのうち, パス長制約が厳しいパスに属す端子同士が, より優先的に同じ部分領域に割り当てられるような次の処理を考える. いま, いくつかの端子がすでに部分領域 R_1, R_2に割当て済みであるものとする. 端子 t_i, t_j間を結ぶパスで, 最もパス長制約の厳しいパスを $t_i \rightarrow t_j$[4]とし, 割当ての済んでいない端子 $t_i \in T_D(R)$ の, 部分領域 $R_k (k = 1, 2)$ に対する依存度 $dep_k(t_i)$ $(k = 1, 2)$ を次のように定義する[5].

$$dep_k(t_i) = \frac{\sum_{t_j \in T(R)} w(t_i \rightarrow t_j) \times [dep(t_i, t_j) \cdot a_k(t_j)]}{\sum_{t_j \in T(R)} w(t_i \rightarrow t_j) \times dep(t_i, t_j)}$$

[3]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.

[4]端子 t_iから端子 t_jに至るパスを $t_i \rightarrow t_j$と書く.

[5]この定義は, t_iが部分領域 Rの入力となる場合である. t_iが部分領域 R の出力となる場合も同様に定義できる.

Step 1. $dep(t_i, t_j)$ $(t_i, t_j \in T_D(R) \cup T_L^p(R_{aug})) \cup T_R^p(R_{aug})$ 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 7: 端子集合の 2 分割アルゴリズム．

ここで，$T(R) = T_D(R) \cup T_L^p(R_{aug}) \cup T_R^p(R_{aug})$ であり，$w(p)$ は，p をパスとしたとき，

$$w(p) = \frac{1}{slack(p)} + 1$$

と定義する．$w(p)$ はスラックの値に基づいたパスの重みと考えることができ，$slack(p) = \infty$ のとき $w(p) = 1$，$slack(p) = 0$ のとき $w(p) = \infty$ となる．$dep_k(t_i)$ は，$0 \leq dep_k(t_i) \leq 1$ の範囲の値をとり，t_iの推移的ファンアウトとなる端子が部分領域 R_k に割り当てられている割合を，パスの制約の厳しさで加重したものを表す．

図 7 に端子集合の 2 分割アルゴリズムを示す．

3.4 LUT 集合の 2 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，

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

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．制約パスがパス長制約を違反することなく，しかも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．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（サイズ制約と呼ぶ）[6]．

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

ただし，

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

ここで，$N_B(R)$ を部分領域 R内部の基本セル数としたとき，$N_L(R) = 4 \cdot N_B(R)$ は R内部の LUT 数を表している．$a = 1/\lg(N_B(R_k) + 1)$ と定義する．

LUT 集合の 2 分割は，ネットワークフローの考えを用いて実現される．以下，本節の用語を定義する．

$G^R = (V, E)$, $G_{st}^R = (V', E')$: 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を $G^R = (V, E)$ とする．G^R 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[6]．これらに対応する G^Rの節点集合をそれぞれ V_L, V_T, V_Nと書く．

G^Rの枝 $e \in E$は，各ネットに対して図 8 のようにとる [11]．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，

- 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．

[6]すぐ後で述べるように，G^R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．

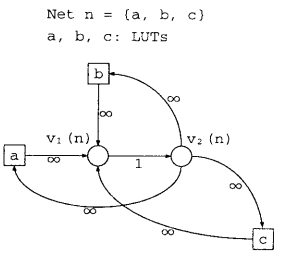


図 8: ネットに対応する容量付きグラフ．

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．

さらに，G^R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_{st}^R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_{st}^R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 $(X, \overline{X})$ が与えられたとき，そのカットサイズ．$sz(X, \overline{X}) = \sum_{e \in E_f(X, \overline{X})} c(e)$.

$V_L(X), V_L(\overline{X})$: 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．

提案する LUT 集合の 2 分割アルゴリズムは，文献 [6] の基本アルゴリズムを継承する．以下，まず，文献 [6] で提案された LUT 集合の基本 2 分割アルゴリズムを簡単に紹介し，パス長制約の導入による基本アルゴリズムの拡張を提案する．

3.4.1 LUT 集合の基本 2 分割アルゴリズム[6]

容量付きグラフ G_{st}^R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_{st}^Rのカット $(X, \overline{X})$ を，サイズ制約を満たし，かつカットサイズが小さくなるように次のように探索する（図 9 参照）．

(1) G_{st}^R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．

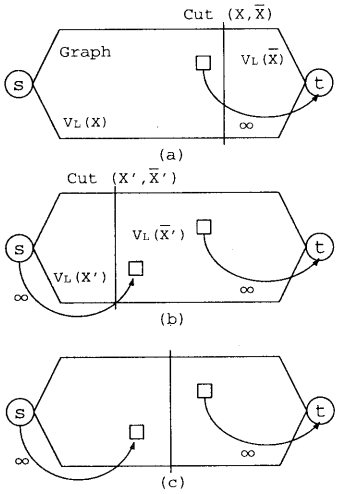


図 9: 最小カットの繰り返し探索．(a) 1 回目の反復．(b) 2 回目の反復．(c) 3 回目の反復（サイズ制約の満足）．

(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 $u \in V_L(\overline{X})$ を接続する．(1) へ．

(3) において，信号が $V_L(X)$ の方から $V_L(\overline{X})$ の方へ向かう流れとなるよう u あるいは vを選択する [6]．この処理により，信号の流れと LUT の入出力端子位置の方向と合致する．

3.4.2 パス長制約を導入した LUT 集合の 2 分割アルゴリズム

LUT 集合の基本 2 分割アルゴリズムを拡張しパス長制約の満足を試みる．次の戦略をとる．

(i) LUT 集合を 2 分割する前に，制約違反を起こすことなしに，各制約パス p が G^R_{st}のカットと交差可能な回数を見積もる（この回数を $nc(p)$ とする）．

(ii) 各制約パス p とカットとの交差回数が $nc(p)$ を越えないような G^R_{st}のカットを探索し，LUT 集合を分割する．

このうち (ii) は，文献 [7] の手法を適用できる．例えば，$nc(p) = 0$ なるパス p を考える（図 10 参照）．LUT 集合の基本 2 分割アルゴリズムにおいて，i 回目の反復までに p を構成する LUT a および d がソース節点 s と接続したとする（図 10(a)）．このとき，もし，LUT b がシンク節点 t と接続すれば，パス p はカットと 2 回以上必ず交差し制約に違反する．これを回避するため，文献 [7] では，$i+1$ 回目の反復の前に，ソース節点と LUT b, c とを容量 ∞ の枝で接続する．パス p は，カットと交差することはなく，$nc(p) = 0$ を満足する[7]．

一方，(i) は，$slack(p)$ の値だけから単純に見積もることはできず，これまでの分割処理で制約パス p 上の LUT，仮想端子が割り当てられた部分領域から見積もる必要がある．以下，(i) に焦点をあて，制約パス p に対し $nc(p)$ を見積もる手法を提案する．

部分領域 R内部のパスは，図 11 にあるような 4 種類に分類することができる．このうち，図 11(b),(c),(d) の場合は，図 11(a) と同様に処理することができるため，以下では，図 11(a) に関して議論する．

[7]なお，文献 [7] の手法では，サイズ制約があるため必ずしも交差数を $nc(p)$ 以下に抑えられるとは限らない．

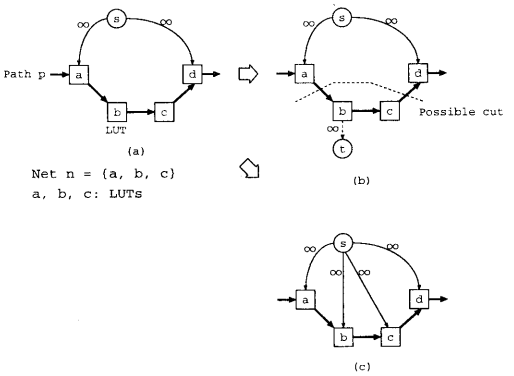


図 10: パス p を構成する LUT のうち a および d がソース節点 s と接続している (a)．LUT b がシンク節点 t と接続した場合，パス p はカットと 2 回交差する (b)．LUT b, c がソース節点 s と接続した場合，パス p はカットと交差しない (c)．

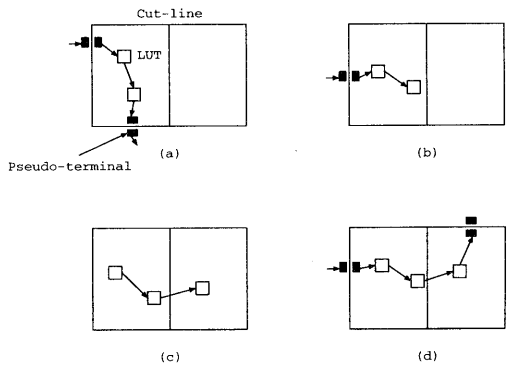


図 11: 部分領域内部でのパスの分類．(a) 始点，終点が共に部分領域の境界にあり，しかもカットラインと同じ側にある場合．(b) 始点あるいは終点の一方が部分領域の境界にある場合．(c) 始点，終点が共に部分領域の内部にある場合．(d) 始点，終点が共に部分領域の境界にあり，しかもカットラインに対し異なる側にある場合．

制約パス p の始点，終点が共にカットラインの左側の部分領域 R_1の境界にある場合を考える．この場合，制約パス p はカットと偶数回交差する．いま，パス p の始点および終点が共に R_1の左辺に割り当てられていると仮定する．図 12(a) のようにパス p がカットと 2 回交差すると，パス長は少なくとも R_1の幅の 2 倍すなわち $2 \cdot w(R_1)$ だけ増加する．その後，さらにカットと 2 回交差する毎にパス長は少なくとも 2 だけ増加する．したがって，パス長制約を満足するためには，パス p はカットと高々

$$nc(p) = slack(p) - [2 \times (w(R_1) - 1)]$$

回のみ交差することができる．仮想端子が左辺に割り当てられている場合だけでなく，制約パス p が始点あるいは終点として R_1の上辺（あるいは下辺）に仮想端子を持ち，p が R_1よりも上（あるいは下）にある部分領域の左辺を通過する場合も上述の議論が適用できる（図 12(b)）．図 11(a) の制約パスに対し，上述以外のパス p では $nc(p) = slack(p)$ と設定する．

以上のように，各制約パス p に対して $nc(p)$ を設定した後，文献 [7] による最小カット探索を実行することにより，パス長制約

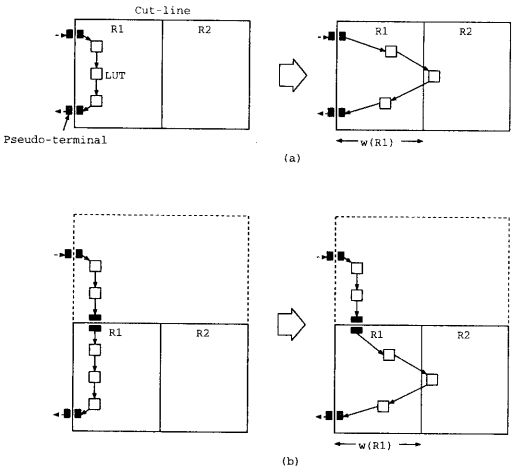


図 12: カットと 2 回交差するパスの例. (a) 始点および終点が部分領域の左辺に存在する場合. (b) 始点が部分領域の上辺に、終点が部分領域の左辺にある場合.

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, $L(R_1) = V_L(X)$, $L(R_2) = V_L(\overline{X})$ として, 終了.

Step 3. 次のどちらかの処理を実行する.

3.1 $|V_L(X)| > M_1$のとき, 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. 文献 [7] に基づき, パス長制約を違反しないよう, サイズ制約の範囲内で制約パス p を構成する LUT と t とを接続する. そのような LUT が存在しない場合, 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. 文献 [7] に基づき, パス長制約を違反しないよう, サイズ制約の範囲内で制約パス p を構成する LUT と s とを接続する. そのような LUT が存在しない場合, 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 13: パス長制約を導入した LUT 集合の 2 分割アルゴリズム.

を満足する LUT 集合の分割が得られることが期待される. 図 13 にパス長制約を導入した LUT 集合の 2 分割アルゴリズムを示す.

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. 通信処理用 FPGA チップ PROTEUS[8] に対し提案手法を適用した. 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[3][8] により論理合成し, PROTEUS 専用 CAD (PROTEUS-CAD) [8] 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. 外部入出力端子は, mpa_ty2 を除いて, 左辺の I/O ブロックに入力端子, 右辺の I/O ブロックに出力端子を割り当てるとし, mpa_ty2 は, 予め設計者によって決められた I/O ブロック位置に外部入出力端子を割り当てるものとする.

[8]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致します.

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

文献 [1],[5],[8] を参考に, パス長制約を次のように設定した.

(1) パス長制約を導入しない従来手法 [6] により回路を配置・概略配線する. 得られた結果から配置結果のみ取り出し[9], PROTEUS-CAD に含まれる配線ツール proroute により配線する.

(2) 遅延が最大となっている信号パスを抽出し, その遅延を d^c_{max}, パス長を l^c_{max}とする. d^c_{max}は回路遅延である.

(3) 遅延が $0.65 \times d^c_{max}$以上となっている全信号パスの集合を P としたとき, パス長制約を次のように設定する.

$$l(p) \leq l_{max}(p) = 0.85 \times l^c_{max} \text{ for all } p \in P$$

表 2 にパス長制約を導入した配置・概略配線結果を示す. #cp はパス長制約を与えたパス数, #v は制約違反を起こしたパス数, l_{max}は制約としたパス長, lは配置・概略配線後の最大パス長を表す. 表から, 提案手法は 2 例を除いて制約違反を 0 にしている. 比較のため, 表 3 にパス長制約を考慮しない従来手法によって配置・概略配線した場合の配線混雑度 (d) および総配線長 (wl) [10] を示す. 提案手法と従来手法とを比較すると, パス長制約を満足するため提案手法の配線混雑度が平均 20%増加している. 総配線長は同程度である. CPU time は平均 10%増加している.

パス長制約の導入により, 回路遅延が削減できることを確認するため, 提案手法による配置結果を proroute により配線した. 表 4 に回路遅延 (max. delay) [11] および配線に要した CPU time を示す. 比較のため, パス長制約を導入しない従来手法 [6] および PROTEUS-CAD 内のツールのみによる配線結果を表 5,6 に示した. 表から, 提案手法は配線混雑度を増加しているものの 100%配線を達成でき, しかもパス長制約を導入することにより, 平均 23%回路遅延が減少していることが確認できる. PROTEUS-CAD のみによる結果では, いくつかの回路に未配線が生じている.

以上の結果から, 提案手法はパス長制約の導入により, 1 章で述べた通信処理用 FPGA に求められる要求 (1),(2) を共に満足していることを確認できた.

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拡張し, パス長制約を導入した. 計算機実験により評価した結

[9]PROTEUS-CAD に含まれる配線ツール 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. そのため配置結果のみ取り出し, proroute の入力とした.

[10]総配線長は, 配置・概略配線後に全ネット上に割り当てられた仮想端子数によって表す.

[11]回路遅延は, proroute によって算出された値を示す.

表 2: パス長制約および配線混雑度に関する実験結果（提案手法）.

circuit	#cp	#v	l_{max}	l	d	wl	t [s]
bip24	45	0	52	46	10	1755	3.64
bip24ed	47	0	44	42	8	976	3.47
bip8	6	0	37	27	7	368	0.45
cnt_3	2	0	35	35	5	101	0.08
cnt_4	4	0	37	33	4	137	0.14
cnt_5	13	0	33	33	5	166	0.22
cnt_6	20	0	53	53	6	210	0.36
cnt_8	15	0	41	41	8	280	0.43
cnt_9	9	0	43	41	7	365	0.41
mpa_ty1	110	1	70	84	15	2752	13.28
mpa_ty2	99	0	72	72	16	2925	12.81
osync	39	1	79	91	9	1679	4.65
scr	16	0	59	51	8	458	0.88
total	425	2	655	649	108	12172	40.82

果，提案手法はパス長制約を 2 例を除き満足することができ，回路遅延を平均 23%削減した．

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．また，第 2 著者は財団法人神奈川科学技術アカデミー研究助成金の援助に感謝いたします．なお，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] C.-S. Chen, Y.-W. Tsay, T. T. Hwang, A. C. H. Wu, and Y. -L. Lin, "Combining technology mapping and placement for delay-optimization in FPGA designs," in *Proc. ICCAD-93*, pp. 123–127, 1993.

[2] K. Kawana, H. Keida, M. Sakamoto, K. Shibata, and I. Moriyama, "An efficient logic block interconnect architecture for user-reprogrammable gate array," in *Proc. IEEE 1990 Custom Integrated Circuits Conf.*, pp. 31.3.1 31.3.4, 1990.

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

[4] 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.

[5] N. Togawa, M. Sato, and T. Ohtsuki, "Maple-opt: A simultaneous technology mapping, placement, and global routing algorithm for FPGAs with performace optimization," in *Proc. ASP-DAC'95*, pp. 319–327, 1995.

[6] 戸川望, 佐藤政生, 大附辰夫, "通信処理用 FPGA を対象とした配置・概略配線同時処理手法," 情処研報, 96-DA-80-3, 1996.

[7] 戸川望, 佐藤政生, 大附辰夫, "パス遅延制約を考慮したマルチ FPGA 用回路分割手法," 第 9 回回路とシステム軽井沢ワークショップ論文集, pp. 67–72, 1996.

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

[9] 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.

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

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

表 3: 配線混雑度に関する実験結果（従来手法[6]）.

circuit	d	wl	t [s]
bip24	7	1497	3.23
bip24ed	9	822	2.26
bip8	7	364	0.48
cnt_3	3	104	0.09
cnt_4	5	165	0.14
cnt_5	5	181	0.19
cnt_6	4	298	0.22
cnt_8	4	402	0.47
cnt_9	4	343	0.45
mpa_ty1	11	3020	11.62
mpa_ty2	13	3229	11.27
osync	9	1111	5.17
scr	7	803	1.57
total	88	12339	37.16

表 4: 詳細配線後の遅延に関する実験結果（提案手法）.

circuit	max. delay [ns]	CPU time [s]
bip24	28.5	453
bip24ed	27.5	281
bip8	18.0	103
cnt_3	10.0	51
cnt_4	15.0	48
cnt_5	14.5	55
cnt_6	25.5	70
cnt_8	26.0	80
cnt_9	24.5	79
mpa_ty1	47.5	1082
mpa_ty2	40.0	914
osync	54.0	401
scr	34.5	124
total	365.5	3741

表 5: 詳細配線後の遅延に関する実験結果（従来手法[6]）.

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

表 6: 詳細配線後の遅延に関する実験結果（**profix and proroute in PROTEUS-CAD**）.

circuit	max. delay [ns]	CPU time [s]
bip24	-	727
bip24ed	-	270
bip8	24.5	98
cnt_3	17.5	41
cnt_4	21.5	45
cnt_5	19.0	63
cnt_6	-	83
cnt_8	-	67
cnt_9	30.5	69
mpa_ty1	-	896
mpa_ty2	-	891
osync	-	502
scr	-	135
total	(113.0)	3887