A Simultaneous Placement and Global Routing Algorithm with Path Length Constraints for Transport-Processing FPGAs

Nozomu Togawa Masao Sato Tatsuo Ohtsuki

Department of Electronics and Communication Engineering
Waseda University
3-4-1 Okubo, Shinjuku, Tokyo 169, Japan
Tel: +81-3-3203-4141 (ext.) 73-5716
Fax: +81-3-3203-9184
e-mail: togawa@ohtsuki.comm.waseda.ac.jp

Abstract— 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 algorithm has been implemented and applied to transport-processing circuits compared with conventional approaches. The results demonstrate that the algorithm resolves path length constraints for 11 out of 13 circuits, though it increases routing congestion by an average of 20%. After detailed routing, it achieves 100% routing for all the circuits and decreases a circuit delay by an average of 23%.

I. Introduction

As application-specific field-programmable gate arrays (FPGAs), transport-processing FPGAs have been proposed [4],[8],[9].[1] In contrast with conventional FPGAs as in [10], transport-processing FPGAs have finer granularity of logic functions so that they implement more flexible circuits on them. Since transport data processing has strong direction of signal flows, input terminals are placed on one side of each cell in FPGAs and output terminals are on the opposite side (cell will be defined in Section II). In placement and global routing design for transport-processing FPGAs, the following requirement should be satisfied first.

(1) Because of fine logic granularity, the amount of connections among cells tends to increase. Placement and global routing with smaller routing congestion is required in order to achieve 100% routing.

In addition, the following requirement should be satisfied in order to achieve real-time processing of a large amount of data communicated continuously.

(2) Circuits implemented on transport-processing FPGAs are required to operate with higher operation frequency.

Several layout synthesis algorithms have been proposed for transport-processing FPGAs [6],[8]. In [8], partitioning-based placement and maze routing for minimizing critical path delays are proposed. Those algorithms are, however, difficult to generate layouts with small routing congestion and thus do not satisfy the requirement (1). In [6], a simultaneous placement and global routing algorithm is proposed. Since it evaluates routing congestion directly during placement, it generates layouts with small routing congestion and thus achieves 100% routing easily. The algorithm, however, does not minimize routing delays explicitly and does not satisfy the requirement (2) as it is. In [1],[5], performance-directed synthesis algorithms for conventional FPGAs have been proposed.[2] The algorithm in [1] executes technology mapping and placement simultaneously and attempts to minimize routing delays. Since it estimates routing delays based on only placement, it cannot obtain accurate routing delays. The algorithm in [5] executes performance-directed technology mapping, placement, and global routing simultaneously. Since it is optimized for FPGAs

[1] FPGAs called PROTEUS have been developed for transport processing [4],[8],[9].

[2] As in [6], placement and global routing for transport-processing FPGAs are regarded as technology mapping and layout in a broad sense. We focused on synthesis algorithms for FPGAs in which both technology mapping and layout synthesis can be executed.

ASP-DAC '97
0-89791-851-7$5.00

proposed in [2], it is difficult to apply it to transport-processing FPGAs.

In this paper, we extend the simultaneous placement and global routing algorithm for transport-processing FPGAs in [6] and propose 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 algorithm has been implemented and applied to transport-processing circuits compared with conventional approaches. The results demonstrate that the algorithm resolves path length constraints for 11 out of 13 circuits, though it increases routing congestion by an average of 20%. After detailed routing, it achieves 100% routing for all the circuits and decreases a circuit delay by an average of 23%.

This paper is organized as follows: Section II defines a path length constraint and a placement and global routing problem; Section III proposes a new simultaneous placement and global routing algorithm incorporating path length constraints; Section IV demonstrates experimental results compared with conventional approaches; and Section V gives concluding remarks.

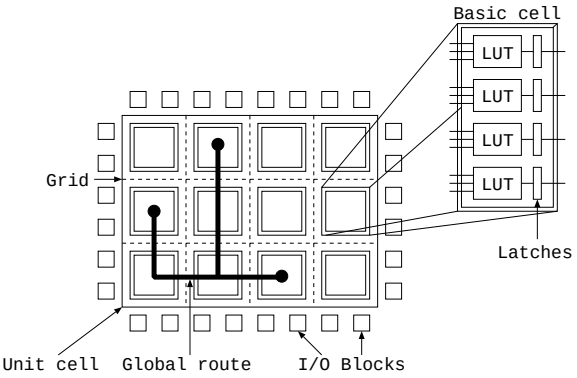
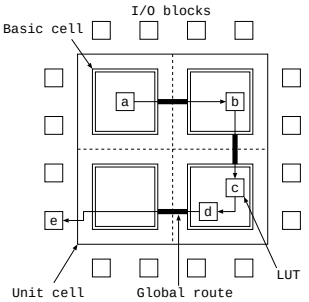


Fig. 1. An FPGA model.



Path $p = a \to b \to c \to d \to e$
Path length $l(p) = 3$

Fig. 2. Path length.

II. Preliminaries

A. FPGA Model

Let us consider an FPGA model depicted as in Fig. 1 based on an FPGA chip proposed in [4],[8],[9]. The model has a two-dimensional array of *basic cells*. Each basic cell contains four LUTs with three inputs and one output as in Fig. 1. An LUT is a unit to realize a logic function. Each LUT in a basic cell has its input terminals on the left side of the basic cell and its output terminals on the right side of the basic cell. The positions of input and output terminals on basic cells determine a signal flow propagated from the left to the right. The output of each LUT may be latched by programming. All the latches in an FPGA chip are connected to a common clock signal with special routing resources. I/O blocks are placed on the four sides of the FPGA model.

B. Path Length Constraints

A *net* is a set of LUTs and primary input and output terminals to be connected. A *netlist* is a set of nets. A global route of each net is represented by a sequence of basic cells (Fig. 1). A *unit-cell* is defined as a minimum room divided by the grid lines on Fig. 1 (each unit-cell contains one basic cell). Routing congestion is defined as the maximum number of global routes which cross a side of a unit cell.

Assume that placement and global routing is given for a netlist. A *signal path* is defined as a path on which a signal given by an LUT or primary input reaches an LUT or primary output via LUTs or directly. A *path length* $l(p)$ of signal path p is defined as the number of sides of unit-cells except for outermost sides through which p passes. For example, Fig. 2 shows an FPGA model composed of 2×2 basic cells and signal path p on which a signal given by LUT a reaches primary output e via LUTs b, c, and d. A path length of p is three, i.e., $l(p) = 3$.

Given signal path p in an input circuit and its upper bound of path length $l_{max}(p)$, a *path delay constraint* is defined as

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

where $l(p)$ is a path length of p after placement and global routing. Path length constraints are imposed on signal paths which are critical for timing.

C. Placement and Global Routing Problem

Now a placement and global routing problem is defined.

Definition 1 *A placement and global routing problem is, for given*

(1) a netlist,

(2) an FPGA model (the numbers of rows and columns of basic cells), and

(3) a set of path length constraints,

to determine

(a) basic cells formed by LUTs,

(b) a placement position of each basic cell, and

(c) a global route of each net

so as to minimize routing congestion under the constraints of

- *used basic cells are accommodated in the given FPGA model,*
- *each basic cell contains at most four LUTs,*
- *primary input and output terminals are assigned to I/O blocks, and*
- *path length constraints.*

Note that, since forming a basic cell by four LUTs is regarded as technology mapping, the above problem determines technology mapping, placement, and global routing simultaneously in a broad sense.

III. A Simultaneous Placement and Global Routing Algorithm Incorporating Path Length Constraints

A. Basic Algorithm

The proposed algorithm is an extended version of the placement and global routing algorithm in [6] and inherits its basic algorithm. Fig. 3 shows the basic algorithm under the following terminology.

subregion: A rectangular region composed of adjacent unit-cells.

cut-line: A horizontal or vertical line bipartitioning one subregion into two pieces. It is drawn along the grid of Fig. 1.

pseudo-terminal: A fictitious terminal placed on a cut-line to preserve the connection of a net divided by the cut-line.

$\boldsymbol{L(R)}$**:** A set of LUTs assigned inside subregion R.

$\boldsymbol{T_L(R), T_R(R), T_U(R), T_D(R)}$**:** Sets of primary input and output terminals or pseudo-terminals assigned on left, right, upper, and lower sides of subregion R, respectively.

$\boldsymbol{w(R), h(R)}$**:** Width and height of subregion R, respectively. A unit is a length of unit-cells.

The basic algorithm is based on (a) hierarchical bipartitioning of layout regions and LUT sets and (b) introducing pseudo-terminals to preserve the connections among

Step 1. Assign primary input and output terminals to four sides of a layout region. Insert the layout region into $\mathcal{Q}$ as a subregion.

Step 2. Pick a subregion R from $\mathcal{Q}$. If $\mathcal{Q} = \emptyset$, stop.

Step 3. Bipartition R into two subregions R_1 and R_2. The cut-line is drawn in such a way that the longer sides of R are partitioned so as to make the partitioned subregions closer to a square.

Step 4. Corresponding to bipartitioning of R,

4.1 bipartition terminal sets on the lower and upper (or right and left) sides of R and

4.2 bipartition an LUT set inside R.

Step 5. If there exists a connection among bipartitioned LUT sets and/or terminals, generate a pair of pseudo-terminals for each net on the cut-line.

Step 6. If R_1 and/or R_2 include more than one unit-cells, insert it (them) into $\mathcal{Q}$. Go to Step 2.

Fig. 3. The basic algorithm.

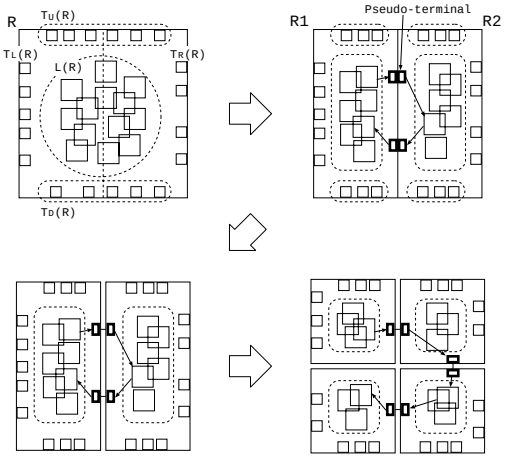


Fig. 4. Recursive bipartitioning of subregions and LUT sets.

bipartitioned LUT sets (see Fig. 4). By repeating hierarchical bipartitioning until each subregion includes just one unit-cell and assigning at most four LUTs to each unit-cell, basic cell formation by LUTs, their placement, and global routing are determined gradually with affecting each other.

The bipartitioning procedure consists of two phases: (1) bipartitioning of a set of terminals[3] and (2) bipartitioning of a set of LUTs. Since those two phases do not consider path lengths explicitly, it is not expected that path length constraints are satisfied.

The proposed algorithm employs the strategy that reduces the lengths of constrained paths by (0) first estimating path lengths and searching the paths with tighter

[3] Pseudo-terminals and primary input and output terminals are referred to terminals for short.

path length constraints and executing phases (1) and (2) extended so that the LUTs and terminals on the paths with tighter path length constraints are assigned to the same subregion with higher priority.[4]

In the rest of this section, Phase (0) and Phases (1) and (2) extended as to incorporate path length constraints are proposed in Sections III.B, III.C, and III.D, respectively.

B. Estimation of Path Lengths

Since the algorithm is based on hierarchical procedures, a signal path is constructed gradually as it proceeds. Path length is equal to the number of pairs of pseudo-terminals on path. Thus a *lower bound* of path length can be computed based on the pseudo-terminals generated so far and their positions. The tightness of a path length constraint for each path is estimated by the difference between the computed lower bound and the upper bound given as a constraint.

A lower bound of length of path p is computed as follows. Assume that path p is composed of k pairs of pseudo-terminals in this level of hierarchy. Among pairs of pseudo-terminals on p, let $PT_v(p)$ be a sequence of pairs of pseudo-terminals assigned to vertical cut-lines, i.e.,

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

where $p_1^v, p_2^v, \ldots, p_m^v$ are in order of a signal flow. Similarly, let $PT_h(p)$ be a sequence of pairs of pseudo-terminals assigned to horizontal cut-lines, i.e.,

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

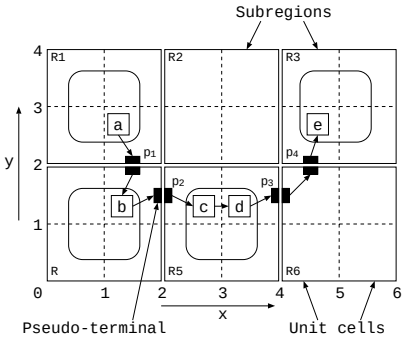
where $p_1^h, p_2^h, \ldots, p_n^h$ are in order of a signal flow. Note $k = m + n$. For two pairs of pseudo-terminals on vertical cut-lines (resp., horizontal cut-lines), if signals flow from the left to the right (resp., from the up to the down) or in its reverse direction for both of them, the directions of the two pairs of pseudo-terminals are called *equal*. Let $x(p_i^v)$ be x coordinate of $p_i^v \in PT_v(p)$ and $y(p_i^v)$ be y coordinate of $p_i^h \in PT_h(p)$ as in Fig. 5. $l_{low}(p)$ is defined as

$$l_{low}(p) = k + \sum_{2 \le i \le m} [|x(p_{i-1}^v) - x(p_i^v)| - d(p_{i-1}^v, p_i^v)] + \sum_{2 \le j \le n} [|y(p_{j-1}^h) - y(p_j^h)| - d(p_{j-1}^h, p_j^h)]$$

where

$$d(p_1, p_2) = \begin{cases} 1 & \left(\begin{array}{c} \text{if the directions of two pairs } p_1 \text{ and} \\ p_2 \text{ of pseudo-terminals are equal} \end{array} \right) \\ 0 & \text{(otherwise)} \end{cases}.$$

[4] The strategy of (0)–(2) employed in the proposed algorithm is the same as that employed in [7]. The algorithm in [7] bipartitions a set of terminals and a set of LUTs based on just connections among them. The proposed algorithm, however, executes Phases (0)–(2) based on geometric information of LUTs and terminals, which will be described in the rest of this section.



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 $p = a \rightarrow b \rightarrow c \rightarrow d \rightarrow e$

Fig. 5. Computation of the lower bound of path length.

$l_{low}(p) \le l(p)$ always holds and $l_{low}(p)$ gives the lower bound of path length of p. For example, let us consider path p in Fig. 5. $p = a \rightarrow b \rightarrow c \rightarrow d \rightarrow e$ has $k = 4$ pairs of pseudo-terminals in this level of hierarchy and $PT_v(p) = \{p_2, p_3\}$, $PT_h(p) = \{p_1, p_4\}$. $l_{low}(p)$ becomes

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

Path p needs four pairs of pseudo-terminals already assigned and at least one pair of pseudo-terminals to pass through R_5.

Let $slack(p)$ be the difference between the lower bound and upper bound of path length, i.e.,

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

The smaller $slack(p)$ is, the tighter path length constraint is imposed on p. A slack of each constrained path is computed before bipartitioning of a set of terminals and bipartitioning of a set of LUTs. A slack of nonconstrained paths sets to be ∞ for simplicity.

C. Bipartitioning of a Set of Terminals

A set of terminals assigned to the upper and lower sides of subregion R are bipartitioned.[5] In this subsection, we discuss bipartitioning of a lower terminal set assuming that a upper terminal set has been already bipartitioned.

A set $T_D(R)$ of terminals assigned to the lower side of R is bipartitioned into two sets $T_D(R_1)$ and $T_D(R_2)$ under the constraints of

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

[5] In Sections III.C and III.D, we discuss bipartitioning with respect to vertical cut-lines.

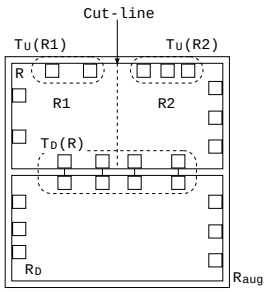


Fig. 6. Augmenting region.

Step 1. Compute $dep(t_i, t_j)$ $(t_i, t_j \in T_D(R) \cup T_L^p(R_{aug}) \cup T_R^p(R_{aug}))$. Let $a_k(t_i) = 0$ and $T_D(R_k) = \emptyset$ $(t_i \in T_D(R),\ k = 1, 2)$.

Step 2. Compute $dep_k(t_i)$ for each unassigned terminal $t_i \in T_D(R)$ and subregion R_k $(k = 1, 2)$.

Step 3. Select terminal t_i whose $dep_k(t_i)$ is maximum. Assign t_i to subregion R_k, i.e., $a_k(t_i) = 1$ and $T_D(R_k) = T_D(R_k) \cup \{t_i\}$. If $|T_D(R_k)| \geq |T_D(R)|/2$, assign all of the unassigned terminals to the other subregion. Otherwise, go to Step 2.

Fig. 7. Bipartitioning algorithm of a terminal set.

$T_D(R_1)$ and $T_D(R_2)$ are assigned to subregions R_1 (the left side of the cut-line) and R_2 (the right side of the cut-line), respectively. In order to keep routing congestion small, $|T_D(R_1)|$ should be equal to $|T_D(R_2)|$. A path with a tight path length constraint is expected to be assigned to the same subregion. The terminals on such path are required to be assigned to the same subregion so that the paths do not cross the cut-line.

Let R_D be a lower adjacent subregion of R. Since $T_D(R)$ is being assigned to the lower side of R and also being assigned to the upper side of R_D, $T_D(R)$ is an interface between R and R_D. In bipartitioning of $T_D(R)$ we must consider region R_{aug} which augments R by adding R_D to it as shown in Fig. 6.

Terminology used in this subsection is defined.

$T_L^p(R_{aug}), T_R^p(R_{aug})$: $T_L^p(R_{aug})$ (resp., $T_R^p(R_{aug})$) is a set of terminals which is assigned to the left (resp., right) side of the cut-line before bipartitioning $T_D(R)$. In Fig. 6, $T_L^p(R_{aug}) = T_U(R_1) \cup T_L(R_{aug})$ and $T_R^p(R_{aug}) = T_U(R_2) \cup T_R(R_{aug})$.

$dep(t_i, t_j)$: A 0-1 variable representing dependency of two terminals $t_i, t_j \in T_D(R) \cup T_L^p(R_{aug}) \cup T_R^p(R_{aug})$. If t_i is a transitive fanin or fanout[6] of t_j, $dep(t_i, t_j)$ is one. Otherwise, it is zero.

$a_k(t_i)$: A 0-1 variable representing whether terminal $t_i \in T_D(R) \cup T_L^p(R_{aug}) \cup T_R^p(R_{aug})$ is included in a bipartitioned subregion R_k $(k = 1, 2)$ or not. $a_1(t_i) = 1 \wedge a_2(t_i) = 0$ for $t_i \in T_L^p(R_{aug})$ and $a_1(t_i) = 0 \wedge a_2(t_i) = 1$ for $t_i \in T_R^p(R_{aug})$. For $t_i \in T_D(R)$, $a_k(t_i) = 1$ if t_i is assigned to subregion R_k $(k = 1, 2)$. Otherwise $a_k(t_i) = 0$.

As described below, the terminals t_i and t_j whose dependency $dep(t_i, t_j)$ is one and which are included in the path with tighter path length constraint are assigned to the same subregion with higher priority.

Assume that several terminals have been already assigned to subregions R_1 and R_2. Let $t_i \rightarrow t_j$ be the path with the tightest path length constraint which connects terminals t_i and t_j.[7] Dependency of unassigned terminal t_i on subregion R_k $(k = 1, 2)$ is defined as

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

where $T(R) = T_D(R) \cup T_L^p(R_{aug}) \cup T_R^p(R_{aug})$ and weight $w(p)$ of path p is defined as

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

$w(p)$ is designed based on a slack. $w(p) = 1$ if $slack(p) = \infty$ and $w(p) = \infty$ if $slack(p) = 0$. $dep_k(t_i)$ represents how much transitive fanouts of t_i are assigned to subregion R_k weighted with tightness of the path delay constraint. A $dep_k(t_i)$ value ranges from 0 to 1. Based on $dep_k(t_i)$, a terminal whose $dep_k(t_i)$ is maximum will be assigned to subregion R_k.

Fig. 7 shows the bipartitioning algorithm of a set of terminals.

D. Bipartitioning of a set of LUTs

A set $L(R)$ of LUTs assigned inside subregion R is bipartitioned into two sets $L(R_1)$ and $L(R_2)$ under the constraints of

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

$L(R_1)$ and $L(R_2)$ are assigned to subregions R_1 (the left side of the cut-line) and R_2 (the right side of the cut-line), respectively. The objective is minimizing the number of nets crossing the cut-line without violating path length constraints. The number of LUTs assigned to each subregion is constrained as follows (*size constraint*) [6]:

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

where

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

[6] If a signal from terminal t_1 reaches terminal t_2 via LUTs or directly, t_2 is called a *transitive fanout* of t_1 and t_1 is called a *transitive fanin* of t_2.

[7] A path from terminal t_i to terminal t_j is denoted as $t_i \rightarrow t_j$.

Here $N_B(R)$ is the number of basic cells in subregion R, $N_L(R) = 4 \cdot N_B(R)$ is the number of LUTs in R, and $a = 1/\lg(N_B(R_k) + 1)$.

A set of LUTs is bipartitioned based on network flows. Terminology used in this subsection is defined.

$G^R = (V, E)$, $G^R_{st} = (V', E')$: G^R is a graph with capacity constructed from a netlist in subregion R. Each node in G^R is associated with either an LUT in R, a terminal on the four sides of R, or a net connecting LUTs and/or terminals in R.[8] Those node sets in G^R are denoted as V_L, V_T, and V_N, respectively.

Edge $e \in E$ of G^R is constructed from each net as in Fig. 8 [11], i.e., for each net n and a set L_n of LUTs and terminals to be connected by n,

- consider two nodes $v_1(n)$ and $v_2(n)$ associated with n,
- connect each $v \in L_n$ to $v_1(n)$ by edge $(v, v_1(n))$ with capacity of ∞,
- connect $v_2(n)$ to each $v \in L_n$ by edge $(v_2(n), v)$ with capacity of ∞, and
- connect $v_1(n)$ to $v_2(n)$ by edge $(v_1(n), v_2(n))$ with capacity of one.

We can apply a network flow technique to bipartition a netlist by introducing the edges described above [11].

Node set V_T is bipartitioned into V_{T_1} and V_{T_2} which are associated with $T_L(R) \cup T_U(R_1) \cup T_D(R_1)$ (the terminals in R on the left side of the cut-line) and $T_R(R) \cup T_U(R_2) \cup T_D(R_2)$ (the terminals in R on the right side of the cut-line), respectively.

G^R_{st} is a graph constructed from G^R, source node s, and sink node t. s is connected to $v \in V_{T_1}$ by edge (s, v) with capacity of ∞ and $u \in V_{T_2}$ is connected to t by edge (u, t) with capacity of ∞. Capacity of edge $e \in E'$ is denoted as $c(e)$.

$(X, \overline{X})$: A partition of nodes in G^R_{st} such that $s \in X$ and $t \in \overline{X}$ and called a *cut*.

$E_f(X, \overline{X})$, $E_b(X, \overline{X})$: Among edges crossing cut $(X, \overline{X})$, $E_f(X, \overline{X})$ is a set of edges whose starting node is in X and their ending node is in $\overline{X}$. $E_b(X, \overline{X})$ is a set of edges whose direction is reverse to $E_f(X, \overline{X})$.

$sz(X, \overline{X})$: $sz(X, \overline{X}) = \sum_{e \in E_f(X, \overline{X})} c(e)$. A cut-size of cut $(X, \overline{X})$.

$V_L(X)$, $V_L(\overline{X})$: Given $(X, \overline{X})$, $V_L(X)$ and $V_L(\overline{X})$ are sets of LUTs included in X and $\overline{X}$, respectively, i.e., $V_L(X) = V_L \cap X$ and $V_L(\overline{X}) = V_L \cap \overline{X}$.

$V_s(X)$, $V_t(\overline{X})$: Given $(X, \overline{X})$, $V_s(X)$ is a set of nodes in X connecting to s and $V_t(\overline{X})$ is a set of nodes in $\overline{X}$ connecting to t.

[8] As described just below, G^R has two nodes associated with one net.

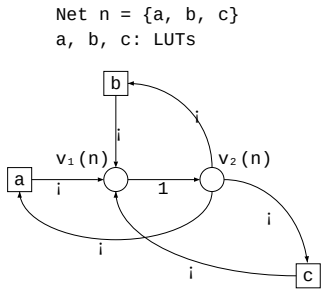


Fig. 8. A graph with capacity for one net.

The proposed bipartitioning algorithm of a set of LUTs inherits the basic algorithm in [6]. First, we introduce the basic algorithm briefly and then extend it so that it incorporates path length constraints.

D.1 Basic Bipartitioning Algorithm of a Set of LUTs[6]

Let us consider a cut $(X, \overline{X})$ in graph G^R_{st}. If $L(R)$ is bipartitioned into $V_L(X)$ and $V_L(\overline{X})$ by the cut, the cut-size of cut $(X, \overline{X})$ is just equal to the number of nets crossing the cut-line of R, which is the same as the number of pairs of pseudo-terminals to be generated, because of the graph representation of nets. The basic algorithm searches a cut $(X, \overline{X})$ in G^R_{st} so that its cut-size is small with satisfying the size constraint in the following way (Fig. 9).

(1) Compute a maximum flow from s to t in G^R_{st} and obtain a minimum cut $(X, \overline{X})$.

(2) If the cut satisfies the size constraint, stop.

(3) Otherwise, connect one node $v \in V_L(X)$ to t if $|V_L(X)| > M_1$ or connect s to one node $u \in V_L(\overline{X})$ if $|V_L(\overline{X})| > M_2$. Go to (1).

In (3), u or v is selected so that a signal flows from $V_L(X)$ to $V_L(\overline{X})$. Thus directions of signal flows are consistent with the positions of terminals on each basic cell.

D.2 Incorporating Path Length Constraints

The basic bipartitioning algorithm of a set of LUTs is extended so that it satisfies path length constraints. The following strategy is employed.

(i) Before bipartitioning a set of LUTs, estimate how many times each constrained path p can cross a cut in G^R_{st} without violating the path length constraint ($nc(p)$ denotes the estimated number of crossings for p).

(ii) Search a cut $(X, \overline{X})$ in G^R_{st} in which each constrained path crosses $(X, \overline{X})$ at most $nc(p)$ times.

Between (i) and (ii), the algorithm in [7] can be applied to (ii). For example, let us consider path p with $nc(p) = 0$ (Fig. 10). Assume that LUTs a and d on p are connected to s when i-th iteration of the basic algorithm is finished

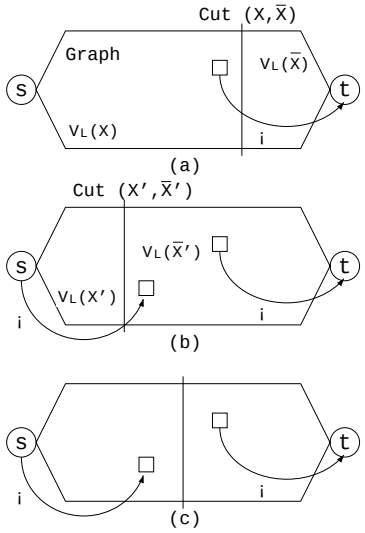


Fig. 9. Repeatedly computed minimum cuts. (a) First iteration. (b) Second iteration. (c) Final iteration in which the size constraint is satisfied.

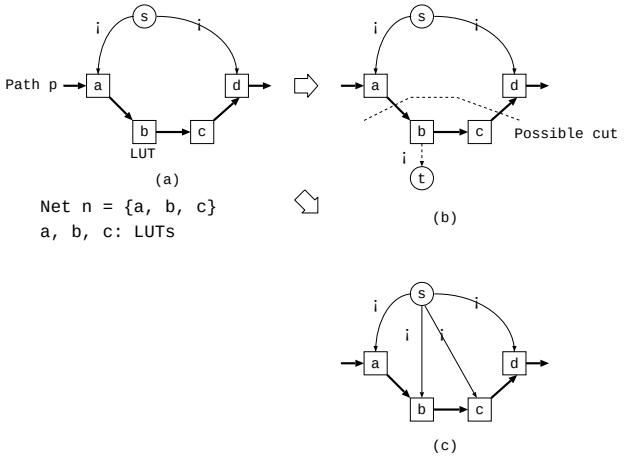


Fig. 10. LUTs a and d on path p is connected to source node s (a). If LUT b were connected to sink node t, p would cross a cut two times (b). If LUTs b and c were connected to source node s, p would never cross a cut (c).

(Fig. 10(a)). If LUT b is connected to t, path p crosses a cut more than one times and violates its path length constraint (Fig. 10(b)). In order not to violate the path length constraint, LUTs b and c are connected to source node s by edges (s, b) and (s, c), respectively, with capacity of ∞ before $(i+1)$-th iteration is started (Fig. 10(c)). Path p does not cross a cut and satisfies $nc(p) = 0$.[9]

In (i), $nc(p)$ can be estimated based on $slack(p)$ and the positions of LUTs and pseudo-terminals generated on p. In the following, we focus on (i) and propose an algorithm to estimate $nc(p)$ for constrained path p.

Paths inside subregion R are classified into four as in Fig. 11 with respect to a vertical cut-line.

Case 1. Paths classified into Fig. 11(a)

Let us consider that both starting and ending points of constrained path p are on region boundary of R_1, subregion in the left side of the cut-line. p crosses a cut even number of times.

Assume that both starting and ending points of p are assigned to the left boundary of R_1. If p crosses a cut two times, path length of p increases by at least $2 \cdot w(R_1)$, where $w(R_1)$ is the width of R_1 as in Fig. 12(a). If p crosses a cut two times more (i.e., four times), its path length increases by at least two more. In order to satisfy the path length constraint of p, p can cross a cut at most

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

times. The above discussion is true when constrained path p has its starting and/or ending points in the upper (resp., lower) boundary of R_1 and p passes through the left boundary of the upper (resp., lower) subregion (see Fig. 12(b)).

Other constrained paths classified into Fig. 11(a) are given $nc(p) = slack(p)$.

Case 2. Paths classified into Fig. 11(b)

Let us consider that either starting or ending point of constrained path p is on region boundary of R_1, subregion in the left side of the cut-line.

Assume that either starting or ending point of p is assigned to the left boundary of R_1. If p crosses a cut one time, path length of p increases by at least $w(R_1)$. If p crosses a cut one more times (i.e., two times), its path length increases by at least one more. In order to satisfy the path length constraint of p, p can cross a cut at most

$$nc(p) = slack(p) - (w(R_1) - 1)$$

times. The above discussion is true when constrained path p has its starting or ending point in the upper (resp., lower) boundary of R_1 and p passes through the left boundary of the upper (resp., lower) subregion.

Other constrained paths classified into Fig. 11(b) are given $nc(p) = slack(p)$.

Case 3. Paths classified into Figs. 11 (c) and (d)

In this case, since path length increases by at least one every time path crosses the cut-line, $nc(p) = slack(p)$ is given.

After $nc(p)$ is computed for each constrained path p, a minimum cut search in [7] is executed so that a set of LUTs is bipartitioned with satisfying path length constraints. Fig. 13 shows the bipartitioning algorithm of a set of LUTs.

[9] As in [7], the number of crossings could be over $nc(p)$ because of the size constraint in the proposed algorithm.

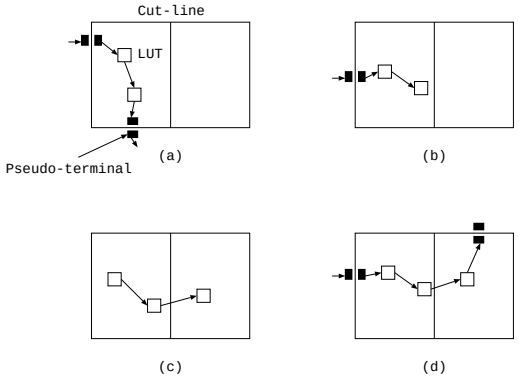


Fig. 11. Pass classification in a subregion. (a) Both starting and ending points of path are on the subregion boundary and they are in the same side of the cut-line. (b) Either starting point or ending point of path is on the subregion boundary. (c) Both starting and ending points of path are inside the subregion. (d) Both starting and ending points of path are on the subregion boundary and they are in the right and left (or upper and lower) sides of the cut-line, respectively.

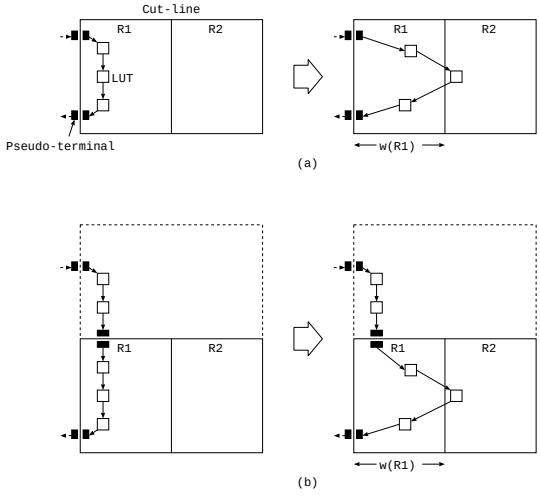


Fig. 12. Paths crossing the cut-line two times. (a) Both starting and ending points of path are on the left side of the subregion. (b) A starting point of path is on the upper side of the subregion and an ending point is on the left side of the subregion.

E. Computational Complexity

Computational complexity is estimated. Since the proposed algorithm is based on hierarchical bipartitioning of subregions, each subregion has a *level* of hierarchy. An entire layout region has the level of one. If a subregion with level of l is bipartitioned, each of bipartitioned subregions has the level of $(l+1)$. Assume that the given number of basic cells is $L_1 \times L_2$ $(L_1 \geq L_2)$. Unit-cells have at most the level of $O(\lg L_1)$. Let N_L be the total number of LUTs and primary inputs and outputs and N_P be the number of generated pseudo-terminals. Let N_N be the number of nets. N_P is at most $O(N_N \cdot (L_1 \times L_2))$. N is defined as $N_L + N_P$. Let N_{cp} be the number of LUTs and pseudo-terminals on paths with path length constraints. If N_c denotes the number of paths with path length constraints, N_{cp} is at most $O(N_c \cdot N)$.

Consider bipartitioning of all subregions with level of l. The most time-consuming procedure is bipartitioning of a set of LUTs. Bipartitioning of a set of LUTs without path length constraints requires $O(N^2)$ time [6]!%By incorporating path length constraints, it requires $O(N_{cp} \lg N_{cp})$ more time [7]. Thus $O(N^2 + N_{cp} \lg N_{cp})$ time is required for bipartitioning of all subregions with level of l.

Based on the above discussion, since there are $O(\lg L_1)$ levels, the time complexity of the proposed algorithm is $O([N^2 + N_{cp} \lg N_{cp}] \cdot \lg L_1)$. The space complexity is $O(N_{cp})$.

IV. Experimental Results

The proposed algorithm has been implemented on SUN Sparc Station 2 (28.5 MIPS) in C language and experimented on PROTEUS, a transport-processing FPGA chip [4],[8],[9]. Table I summarizes the numbers of inputs, outputs, LUTs, and latches (#inputs, #outputs, #LUTs, and #FFs, respectively) of transport-processing circuits to be experimented. To obtain the circuits except for mpa_ty1 and mpa_ty2, we executed a logic synthesizer called PARTHENON [3],[10] for behavioral descriptions (SFL files) followed by a technology mapper called procover in PROTEUS-CAD [8]. Primary inputs and outputs are assigned to I/O blocks on the left and right side of the chip, respectively, except for mpa_ty2. In mpa_ty2, they are assigned to the I/O blocks already specified by a designer.

Based on [1],[5],[8], path length constraints for each experimented circuit are generated as follows:

(1) Execute the conventional simultaneous placement and global routing algorithm in [6] in which path length constraints are not incorporated for each circuit. Pick up the placement result[11] and route it by the routing tool, proroute, in PROTEUS-CAD.

(2) Extract signal paths whose delay is maximum. Let d^c_{max} be the maximum delay and l^c_{max} be the maximum path length of the signal paths. d^c_{max} gives the circuit delay.

(3) Let P be a set of signal paths whose delay is more than or equal to $0.65 \times d^c_{max}$. Path length constraints are given by

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

[10] The authors would like to thank NTT and NTT Data Communications for providing them with PARTHENON.

[11] The routing tool "proroute" in PROTEUS-CAD does not divide routing into global and detailed routing but generates detailed routing results directly from placement results. Thus we used placement results only and gave them to proroute as input.

Step 0. Construct $G_{st}^R = (V', E')$ from a netlist in subregion R.

Step 1. Compute a cut $(X, \overline{X})$ in G_{st}^R whose $sz(X, \overline{X})$ is minimum and $|X|$ is minimum.

Step 2. If $|V_L(X)| \leq M_1 \wedge |V_L(\overline{X})| \leq M_2$, let $L(R_1) = V_L(X)$ and $L(R_2) = V_L(\overline{X})$ and stop.

Step 3. Execute either Step 3.1 or Step 3.2.

3.1 If $|V_L(X)| > M_1$, connect sink node t to each node $u \in \overline{X}$ by edge (u, t) with capacity of ∞. Based on [7], select LUTs in $V_L(X) - V_s(X)$ on constrained paths to be connected to t and connect them to t without violating the size constraint. If there exist no such LUTs, connect t and one node $v \in V_L(X) - V_s(X)$ by edge (v, t) with capacity of ∞. Go to Step 1.

3.2 If $|V_L(\overline{X})| > M_2$, connect source node s to each node $v \in X$ by edge (s, v) with capacity of ∞. Based on [7], select LUTs in $V_L(\overline{X}) - V_t(\overline{X})$ on constrained paths to be connected to s and connect them to s without violating the size constraint. If there exist no such LUTs, connect s and one node $u \in V_L(\overline{X}) - V_t(\overline{X})$ by edge (s, u) with capacity of ∞. Go to Step 1.

Fig. 13. Bipartitioning algorithm of an LUT set incorporating path length constraints.

Table II shows placement and global routing results with path length constraints. #cp and #v denote the number of paths on which path length constraints is imposed and the number of paths violating constraints, respectively. l_{max} and l denote upper bound of path length for constrained paths and the maximum path length after placement and global routing. The table shows that the proposed algorithm resolves the constraints for 11 out of 13 circuits. For comparison, Table III shows routing congestion (rc) and total wire length (wl)[12] for the conventional algorithm in [6] in which path length constrains are not incorporated. Routing congestion of the proposed algorithm increases by an average of 20% in order to satisfy path length constraints. Its total wire length is comparable. Its CPU time increases by an average of 10%.

In order to confirm that circuit delays are reduced by incorporating path length constraints, the placement results obtained by the proposed algorithm are routed using proroute. Table IV shows circuit delay (max. delay) and CPU time required in routing. For comparison, routing results obtained by the conventional algorithm in [6] and the tools only in PROTEUS-CAD are shown in Tables V and VI, respectively. Those tables show that the proposed algorithm achieves 100% routing for all the experimented circuits and decreases a circuit delay by an average of 23%. The results obtained by the tools only in PROTEUS-CAD show that they cause unrouted nets for several circuits.

[12] Total wire length refers to the number of pseudo-terminals assigned to all the nets after placement and global routing.

TABLE I
EXPERIMENTED CIRCUITS.

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

From the above results, we confirm that the proposed algorithm satisfies the requirements (1) and (2) for transport-processing FPGAs described in Section I.

V. CONCLUSIONS

We have proposed a simultaneous placement and global routing algorithm incorporating path length constraints for transport-processing FPGAs. The experimental results demonstrate that the algorithm resolves almost all the path delay constraints and thus reduces a circuit delay by an average of 23%.

ACKNOWLEDGMENTS

The authors would like to thank Dr. N. Ohta, Dr. T. Miyazaki, Dr. A. Tsutsui, and other members of NTT Optical Network Systems Labs. for providing them with PROTEUS-CAD and giving them valuable comments. This work was supported in part by Grant-in-Aid for Scientific Research from the Ministry of Education, Science, and Culture of Japan and by Kanagawa Academy of Science and Technology Research Grants.

REFERENCES

[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 systems labs," *IEICE Trans. Inf. & Syst.*, vol. E76-D, no. 9, pp. 1047–1054, 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.

TABLE II
RESULTS ON PATH LENGTH CONSTRAINTS AND ROUTING CONGESTION (OURS).

circuit	#cp	#v	l_{max}	l	rc	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

#cp: the number of paths on which path length constaints are imposed
#v: the number of paths violating path length constraints
l_{max}: upper bound of path length for constrained paths
l: maximum path length for constrained paths after placement and global routing
rc: routing congestion
wl: total wire length

TABLE III
RESULTS ON ROUTING CONGESTION ([6]).

circuit	rc	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

TABLE IV
RESULTS ON CIRCUIT DELAYS AFTER DETAILED ROUTING (OURS).

circuit	max. delay [ns]	t [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

TABLE V
RESULTS ON CIRCUIT DELAYS AFTER DETAILED ROUTING ([6]).

circuit	max. delay [ns]	t [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

TABLE VI
RESULTS ON CIRCUIT DELAYS AFTER DETAILED ROUTING (profix AND proroute IN PROTEUS-CAD).

circuit	max. delay [ns]	t [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

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

[6] N. Togawa, M. Sato, and T. Ohtsuki, "Simultaneous placement and global routing for transport-processing FPGA layout," *IEICE Trans. Fundamentals*, vol. E79-A, no. 12, 1996.

[7] N. Togawa, M. Sato, and T. Ohtsuki, "A performance-oriented circuit partitioning algorithm with logic-block replication for multi-FPGA systems," in *Proc. APCCAS'96*, 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.