Maple-opt: A Simultaneous Technology Mapping, Placement, and Global Routing Algorithm for FPGAs with Performance Optimization

Nozomu Togawa Masao Sato Tatsuo Ohtsuki

Dept. of Electronics and Communication Engineering
Waseda University
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 this paper, we propose a new FPGA design algorithm, Maple-opt, in which technology mapping, placement, and global routing are executed so that the delay of each critical signal path in an input circuit is within a specified upper bound imposed on it. The basic algorithm of Maple-opt is top-down hierarchical bi-partitioning of regions. Technology mapping onto logic-blocks of FPGAs, their placement, and global routing are determined simultaneously in each hierarchical process. This simultaneity leads to less congested layout for routing. In addition to that, Maple-opt computes a lower bound of delay for each path with a constraint value and determines critical paths based on the difference between the lower bound and the constraint value dynamically in each hierarchical process. Two delay reduction processes are executed for the critical paths; one is routing delay reduction and the other is logic-block delay reduction. Routing delay reduction is realized such that, when bi-partitioning a region, each constrained path is assigned to one subregion. Logic-block delay reduction is realized such that each constrained path is mapped onto fewer logic-blocks. Experimental results for some benchmark circuits show its efficiency and effectiveness.

I. Introduction

In designing FPGAs (Field-Programmable Gate Arrays), one of the greatest concerns is that designers cannot always realize required speed of circuits. Delay by blocks realizing logic functions (logic-blocks) and signal propagation delay by routing are main factors that limit the speed of FPGAs. Between them, the propagation delay by routing has greater influence on circuit speed since routing is realized by switch elements such as pass-transistors or anti-fuses [2],[6],[13]. To consider and reduce the delay by those factors explicitly is essential for designing FPGAs.

The following two points are important in such design.

(1) Reduction of delay by routing and delay by logic-blocks should be considered simultaneously. Therefore, technology mapping and layout design are required to interact with each other.

(2) Most conventional performance-driven layout algorithms divide signal paths into several nets and put restrictions on the length of each net [5],[10]. They are not sufficient, however, since switch elements cause larger routing delay in FPGAs. Delay along signal paths should be handled directly.

The algorithm proposed in [3] aims at reduction of delay during both technology mapping and layout design. However, it executes placement only as layout design and then it is difficult to obtain delay based on accurate routing information. We have proposed Maple, which executes technology mapping, placement, and global routing simultaneously for Lookup Table-based (LUT-based) FPGAs [12]. By executing mapping and layout simultaneously in recursive process, Maple utilizes unused logic-blocks effectively and generates less congested layout. It is insufficient for practical FPGA design, however, since it takes no account of delay reduction. In this paper, we extend Maple and propose an FPGA design algorithm, Maple-opt, in which technology mapping, placement, and global routing are executed so that the delay of each critical signal path in an input circuit is within a specified upper bound imposed on it. In addition to the process of Maple [12], Maple-opt executes the following processes so as to satisfy Points (1) and (2) mentioned above.

- A lower bound of delay of each path with a constraint value is computed in each level of hierarchical process. Critical paths are detected based on the difference between the lower bound and the constraint value, and delay reduction process is executed for the paths.

- Since technology mapping, placement, and global routing are executed in each hierarchical process, Maple-opt can control both logic-block delay and routing delay in constrained paths.
- To reduce routing delay, when partitioning a region into two subregions, Maple-opt assigns nodes in each critical path to the same subregion.
- To reduce logic-block delay, Maple-opt is executed so that nodes in each critical path are covered with fewer logic-blocks.

This paper is organized as follows: Section II defines path delay constraints and a technology mapping, placement and global routing problem; Section III describes the proposed algorithm; Section IV demonstrates experimental results compared with conventional approaches; and Section V gives concluding remarks.

II. Path Delay Constraints and Problem Formulation

A. Preliminaries

In this paper, a symmetric layout model depicted as in Fig. 1 is employed as an FPGA [6],[12]. Each *logic-block* is connected with *switch-blocks* through terminals. Logic-blocks on the periphery of the layout model are *I/O blocks* and each of them has input and output terminals for all of the adjacent switch-blocks. Other logic-blocks contain LUTs and realize any combinational logic up to four inputs. They are connected with adjacent switch-blocks as in Fig. 2. Switch-blocks connect among tracks and terminals. The position for placing a logic-block is called a *slot* and a set of tracks which connects two adjacent switch-blocks is called a *channel*.

The input of Maple-opt is a Boolean network. If the number of incoming edges to each node of a Boolean network is less than or equal to four, the network is called *feasible*. For a node s in a Boolean network, a set of s and transitive fanins [1] of s is called a *cover of* s. If the number of nodes which have outgoing edges to a cover of s is less than or equal to four, the cover is called *feasible*. Each feasible cover of s is realized by one logic-block. Technology mapping is to obtain covers for an input feasible boolean network such that (a) each of primary inputs and outputs corresponds to one logic-block (cover), (b) each of other nodes belongs to at least one cover, and (c) inputs of any cover are outputs from other covers or primary inputs.

B. Path Delay Constraints

Delay of signal paths in an FPGA is composed of delay by logic-block and delay by routing among logic-blocks [2],[6],[13].

[1] If there exists a directed path from s to t, s is called a *transitive fanin of* t and t is called a *transitive fanout of* s.

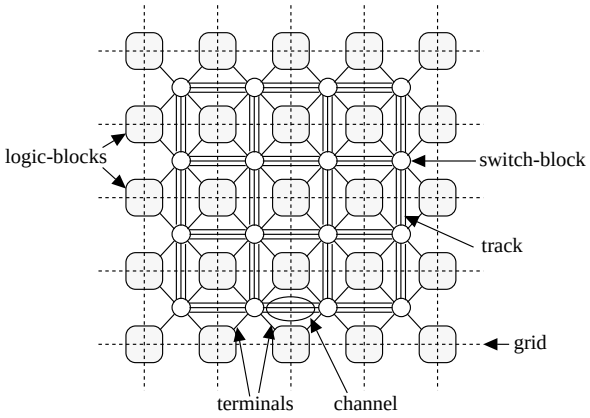


Fig. 1. FPGA model.

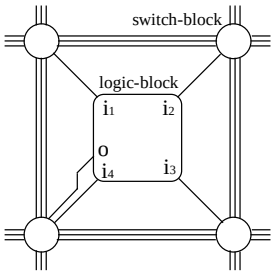


Fig. 2. Terminal positions of a logic-block (i_1, i_2, i_3, i_4: input terminals, o: output terminals).

Since an LUT can be viewed as a memory, logic-block delay is given by a constant L_d.

Routing delay depends on the number of switch elements connecting tracks, which is determined by the number of switch-blocks in a signal path. Let S_d be delay by one switch-block. Let L_n be the number of logic-blocks and S_n be the number of switch-blocks in a signal path. Path delay $d(p)$ for a signal path p is defined as

$$d(p) = L_n \times L_d + S_n \times S_d$$

Note that start and end logic-blocks in a path are excluded in L_n. Fig. 3 shows an example of path delay where a signal propagates a path from one logic-block x to the other w. Given a signal path p and its upper bound of path delay $d_{max}(p)$, *a path delay constraint* is defined as $d(p) \leq d_{max}(p)$.

In FPGA design, the numbers of logic-blocks and switch-blocks in a path are determined by technology mapping and global routing, respectively. Therefore, if a simultaneous technology mapping, placement, and global routing process is executed so as to reduce the numbers of logic-blocks and switch-blocks in each constrained path, the path delay must be maintained within its upper bound.

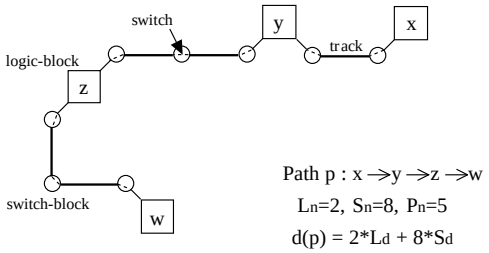


Fig. 3. Path delay.

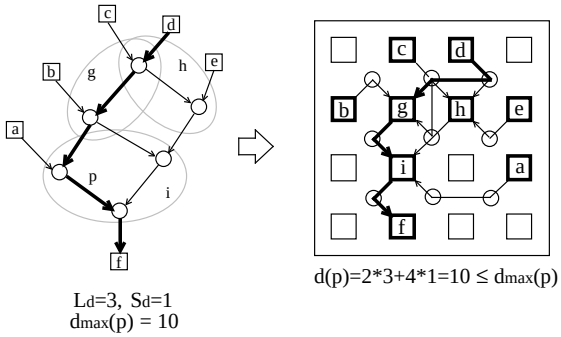


Fig. 4. Boolean network with a path delay constraint and its layout.

C. Problem Formulation

We consider the following problem by adding path delay constraints to the problem defined in [12].

Definition Technology mapping, placement, and global routing problem is, for given

(1) a feasible Boolean network,

(2) slots and terminal positions of logic-blocks, and

(3) path delay constraints,

to determine

(a) covers for the input Boolean network (technology mapping),

(b) a slot position where each logic-block is placed, with considering each cover to be one logic-block (placement),

(c) a terminal position where each terminal of each net is connected (pin assignment), and

(d) channels through which each net passes (global routing)

so as to satisfy the path delay constraints and minimize channel density (the maximum number of tracks per channel for global routing) within given slots. Note that I/O blocks are needed to be placed on the outermost slots.

Fig. 4 shows an example of an input Boolean network with a path delay constraint and its layout.

III. The Maple-opt Algorithm

A. Basic Algorithm

Maple-opt is an extended version of Maple [12] and inherits the algorithm of Maple as a basic algorithm. Therefore, it has basic performance of Maple. Fig. 5 shows the basic algorithm (Maple) under the following terminology.

Unit-cell: A room partitioned by the grid (broken lines) of Fig. 1.

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

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

Pseudo-block: A fictitious block placed on a cut-line to maintain the connection of a net divided by the cut-line. Every pseudo-block is assigned in a channel.

Block set: A set of logic-blocks and pseudo-blocks. Each block set is assigned on a subregion boundary.

Node set: A set of nodes of a Boolean network. Each node set exists inside a subregion.

The algorithm is based on recursive bi-partitioning of subregions in the same way as the min-cut placement method [1] (cf., Fig. 6). In each recursive bi-partition, technology mapping (Step 6), partitioning of block sets and node sets, and their assignment to subregions (Steps 4 and 7) are executed. Those steps proceed with affecting each other. Finally, we obtain logic-blocks generated by technology mapping, their placement, and global routes represented by sequences of pseudo-blocks.

B. Incorporation of Path Delay Constraints

Now, let us consider how to extend the basic algorithm described in the previous subsection to satisfy path delay constraints. We employ the strategy that reduces path delay in each recursive process by (1) detecting paths with tight path delay constraints (which we call critical paths), and (2) reducing the numbers of logic-blocks and switch-blocks in the detected paths.

Let L_n and T_n be the number of logic-blocks and the number of tracks in a path, respectively. Then, the number of switch-blocks S_n in the path is written by

$$S_n = L_n + T_n + 1$$

Step 1. Assume each of primary inputs and outputs of an input Boolean network to be an I/O block. I/O blocks are placed on four sides and four corners of the layout region. Put the entire layout region as a subregion into a queue Q.

Step 2. Pick a subregion R from Q. If Q is empty, halt.

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

Step 4. Corresponding to the partition of R, partition each of two block sets placed on vertical (horizontal) sides of the subregion boundary into three sets. Then, assign each of sets to the upper or lower (left or right) side of the cut-line, or on the cut-line (cf., Fig. 6).

Step 5. For nodes inside R, let candidate nodes for technology mapping be the nodes which are expected to be assigned on the cut-line.

Step 6. Execute covering process for the candidate nodes (technology mapping).

Step 7. Place the covers generated by Step 6 on the cut-line (each cover becomes one logic-block). Assign each uncovered node inside R to the upper or lower (left or right) side of the cut-line.

Step 8. For the logic-blocks on the cut-line, assign nets to their terminals.

Step 9. Generate pseudo-blocks on the cut-line. If R_1 and/or R_2 can be further partitioned, put them (it) into Q. Return to Step 2.

Fig. 5. Basic algorithm.

Therefore, reduction of S_n can be equivalent to reduction of both L_n and T_n. Since each pseudo-block corresponds to a track, reduction of T_n means reduction of pseudo-blocks inserted in a path. From the above discussion, in order to satisfy path delay constraints, we must develop an algorithm which reduces the numbers of logic-blocks and pseudo-blocks generated in each recursive bi-partitioning, and apply it to critical paths with higher priority.

In the rest of this section, we discuss detection of critical paths (Section III-C), reduction of pseudo-blocks (Section III-D), and reduction of logic-blocks (Section III-E).

C. Detection of Critical Paths

Since Maple-opt is based on a hierarchical process, elements which compose a path (logic-blocks and switch-blocks) are determined gradually as it proceeds. Based on those elements, a lower bound of delay in each level of hierarchy can be computed for each constrained path.

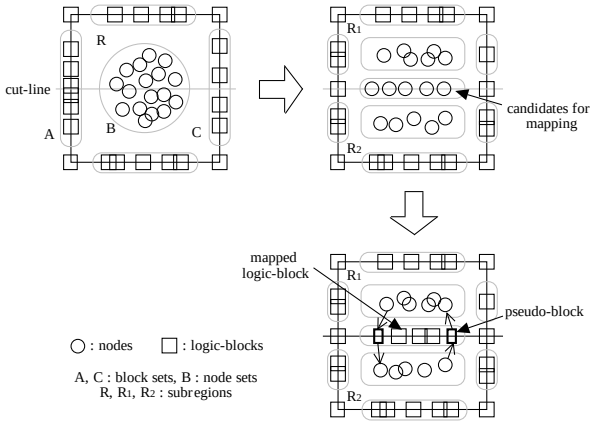


Fig. 6. Recursive bi-partition of subregions.

We detect paths with tighter path delay constraints dynamically in every hierarchy by computing the difference between a lower bound and a constraint value.

The lower bound of path delay is computed as follows. Let $L_{nb}(p)$ and $P_{nb}(p)$ be the number of logic-blocks and the number of pseudo-blocks which have already been generated in a path p so far in the recursive process. Whenever uncovered nodes of a Boolean network exist between a pair of the generated blocks in p, at least one logic-block is necessary to cover those nodes (Fig. 7). In other words, if $B(p)$ is the number of such parts in p, at least $B(p)$ logic-blocks will be generated besides already generated $L_{nb}(p)$ logic-blocks. Then, $d(p)$, i.e. path delay of p after global routing satisfies

$$d(p) \geq d_{low}(p) = [L_{nb}(p) + B(p)] \cdot L_d + [P_{nb}(p) + [L_{nb}(p) + B(p)] + 1] \cdot S_d$$

where $d_{low}(p)$ is the lower bound of path delay in p in this level of hierarchy.

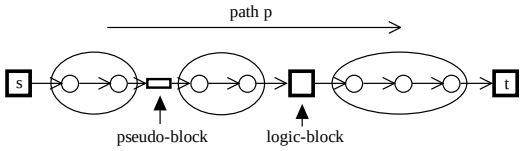
Using a delay constraint value $d_{max}(p)$ and its lower bound $d_{low}(p)$, $slack(p)$ is defined as

$$slack(p) = d_{max}(p) - d_{low}(p)$$

The smaller $slack(p)$ is, the tighter path delay constraint imposed on p is. Paths with small slacks are regarded as critical paths. $slack(p)$ is computed every time when picking a subregion (Step 2).

D. Reduction of Pseudo-Blocks

The number of pseudo-blocks generated in each recursive process is determined in the partition of block sets and node sets and their assignment to subregions (Steps 4 and 7). Since block sets can be treated in the same way as node sets in the partition and assignment process, a node set is focused on in this subsection with respect to a horizontal cut-line.



$L_{nb}(p) = 1$
$P_{nb}(p) = 1$
$B(p) = 3$
$d_{low} = [1 + 3]*L_d + [1 + (1+3) + 1]*S_d$

Fig. 7. Computation of a lower bound of path delay.

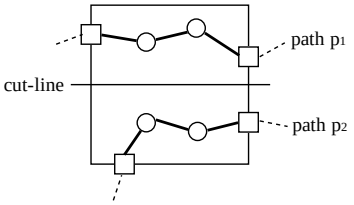


Fig. 8. Path classification for reduction of pseudo-blocks.

Nodes inside a subregion are partitioned based on labels associated with them. A label of a node v, $label(v)$ ($0 \leq label(v) \leq 1$), is designed such that the closer to 1 (resp., 0) $label(v)$ is, the more likely v is assigned to the upper (resp., lower) side of the cut-line. The label of each node is determined in Step 5, i.e. selection of candidate nodes for technology mapping. In Maple [12], the labels reflect only overall connection between nets. Those labels are, however, insufficient for satisfying path delay constraints. In Maple-opt, we compute $label(v)$ for a node v by adding the conventional label reflecting overall connection between nets ($label_c(v)$) and a new label aiming at satisfaction of path delay constraints ($label_p(v)$).

In order to reduce the number of generated pseudo-blocks, nodes in a constrained path should be assigned to the same side of the cut-line. By adding $label_p$ to $label_c$, the labels of those nodes should be kept in as equal a value as possible according to tightness of the constraints.

In Maple-opt, if all of the nodes inside a subregion and the blocks on subregion boundary which compose a path p can be assigned to the upper side of the cut-line (in case of a path p_1 in Fig. 8) or to the lower side of the cut-line (in case of a path p_2 in Fig. 8), $label_p(v)$ for v ($\in p$) is computed as

$$label_p(v) = \begin{cases} +1/[slack(p)+1] \\ \quad \text{(in case } p \text{ can be assigned to the upper side)} \\ -1/[slack(p)+1] \\ \quad \text{(in case } p \text{ can be assigned to the lower side)} \end{cases}$$

Otherwise, $label_p(v) = 0$.

If a node v belongs to more than one constrained paths, $label_p(v)$ is computed for the tightest constrained path.

The smaller $slack_p(v)$ is, the closer to 1 (or -1) $label_p(v)$ becomes and then the more it affects $label(v)$. When $slack(p) = 0$, $label(v)$ for a path p is set to be 1 (or 0) and nodes in p are forced to be assigned to the upper (or lower) side of the cut-line.

Fig. 9 shows an overview of selection process of candidate nodes for technology mapping (Step 5) in which the labels of nodes are computed. Based on the label values which include consideration of path delay constraints

Step 5. Computation of labels and selection of candidate nodes for technology mapping.

(1) For each node v inside the given subregion, compute $label_c(v)$.

(2) For each node v, compute $label_p(v)$.

(3) For each node v, compute $label(v)$ as follows:

$$label(v) = \begin{cases} \min\{label_c(v) + label_p(v), 1\} & (\text{if } label_p(v) \geq 0) \\ \max\{label_c(v) + label_p(v), 0\} & (\text{if } label_p(v) < 0) \end{cases}$$

(4) Sort the nodes inside the subregion in the descending order of $label(v)$ and assign them to the upper side of the cut-line, on the cut-line, and to the lower side of the cut-line in proportion to the number of slots in their area. Let nodes assigned on the cut-line be candidate nodes for technology mapping.

Fig. 9. Selection of candidate nodes for technology mapping for the purpose of reducing pseudo-blocks.

computed in Step 5, the partition of a node set and assignment to subregions are executed in Step 7.

E. Reduction of Logic-Blocks

The number of logic-blocks generated in each recursive process is determined by Step 6, i.e. covering process for the candidate nodes [2]. First, terminology used in this subsection is defined.

N_s, B_s: A node set inside a subregion R and a block set on region boundary, respectively.

N_{cand}: A candidate node set for technology mapping. Let $l_{min} = \min_{v \in N_{cand}} label(v)$, and $l_{max} = \max_{v \in N_{cand}} label(v)$.

$G(t)$: A graph which is composed of t ($\in N_{cand}$), transitive fanins of t (nodes and blocks) in a subregion R, and a fictitious node s (start node). The start node s is connected to blocks in $G(t)$ as in Fig. 10(a).

$N_{G(t)}$: A node set of $G(t)$.

[2] Some non-candidate nodes could be covered in order to obtain feasible covers.

$N_p(u \rightarrow v)$: A node set from u to v in a path p. We assume $u, v \in N_p(u \rightarrow v)$.

$G_R(t)$: A graph with capacity generated from $G(t)$ (see Fig. 10(b)). For each node v in $G(t)$ besides s and t, two nodes v_1 and v_2 are associated with v and connected by a directed edge (v_1, v_2) with capacity of one in $G_R(t)$. Incoming edges of v correspond to those of v_1 and outgoing edges of v correspond to those of v_2. Those edges have capacity of ∞ in $G_R(t)$.

$N^s_{G_R(t)}$: A node set including s when bi-partitioning $G_R(t)$.

$N^t_{G_R(t)}$: A node set including t when bi-partitioning $G_R(t)$.

Nodes in each constrained path are required to be covered with as few logic-blocks as possible. Maple-opt covers candidate nodes by computing a cut in $G_R(t)$ of a maximum-flow and minimum-cut technique repeatedly. At that time, it executes covering such that one cover includes more nodes in the constrained path.

For instance, let us consider covering of a candidate node t for mapping with respect to an LUT with three inputs. Figs. 10(a) and (b) show $G(t)$ and $G_R(t)$, respectively. Bold lines in Fig. 10(a) indicate a constrained path. For simplicity, we assume $N_{cand} = N_s$.

According to the maximum-flow and minimum-cut theorem [9], a maximum flow from s to t in $G_R(t)$ gives a minimum cut with the size of two (Fig. 10(b)). Then, $cover(t)$ is defined as

$$cover(t) = \{t\} \cup \{v \in N_{G(t)} \mid v_1, v_2\ (\in N_{G_R(t)}) \text{ associated with } v \text{ s.t. } v_1 \in N^t_{G_R(t)} \wedge v_2 \in N^t_{G_R(t)}\}$$

The number of inputs for $cover(t)$ is equal to the cut size and $cover(t)$ gives a feasible cover to t. In Fig. 10(b), the number of inputs for $cover(t)$ is two.

In this case, since the number of inputs for the cover is less than three, the cover could include more nodes with maintaining its feasibility. Maple-opt augments an amount of a flow in $G_R(t)$ so that $cover(t)$ can include more nodes in a path with a tight path delay constraint. Let v be a node in p with the tightest constraint satisfying

$$N_p(v \rightarrow t) \subseteq N_{G(t)} \ \wedge\ v_1, v_2\ (\in G_R(t)) \text{ associated with } v \text{ s.t. } v_1 \in N^s_{G_R(t)} \wedge v_2 \in N^t_{G_R(t)}$$

Maple-opt selects the node v and augments an amount of the flow from s to v_1 in $G_R(t)$. Finally, the flow from s to v_1 is filled and its amount becomes three. v_1 is included in $N^t_{G_R(t)}$ and $cover(t)$ satisfies

$$N_p(v \rightarrow t) \subseteq cover(t)$$

We obtain one logic-block which covers a path from v to t as in Figs. 10(c) and (d). The tighter the constraint of a path is, the fewer logic-blocks the path is expected to be covered with, and therefore delay for the path is expected to be reduced.

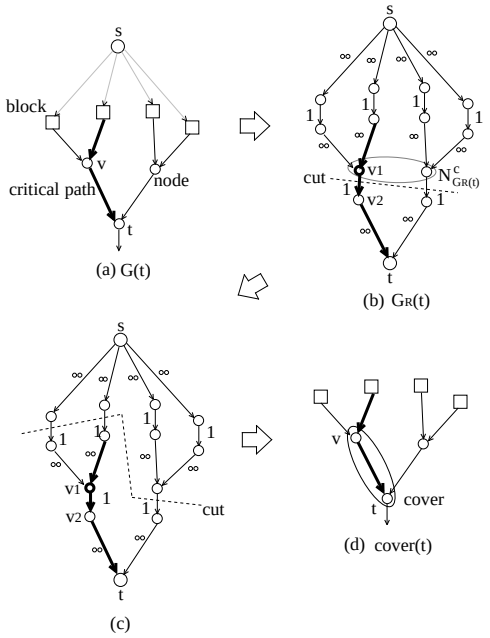


Fig. 10. Covering a critical path.

Fig. 11 shows a covering process for candidate nodes for technology mapping. Note that we use the augmenting path method [9] in order to compute a maximum flow and Steps (7) and (8) in Fig. 11 are executed in the same way as in [12].

F. Computational Complexity

Since Maple-opt is based on top-down hierarchical bi-partitioning, each subregion has a *level* of the hierarchy. The level of the entire layout region is one. If a subregion with level of i is bi-partitioned, each of bi-partitioned subregions has the level of $(i+1)$. If the number of slots on an FPGA chip is $L \times L$, the level of unit-cells is $O(\log L)$. Let N_n, N_{cp}, and N_p be the number of nodes of a Boolean network, the number of constrained paths, and the number of pseudo-blocks generated in the whole process, respectively. If T is the maximum number of tracks per channel, N_p is at most $O(L^2 \cdot T)$. Let N be $N_n + N_p$.

At each level of hierarchy, time complexity without consideration of path delay constraints is $O(N \cdot (\log N + N_n))$ [12]. In addition to that, both reduction process of pseudo-blocks and that of logic-blocks require $O(1)$ time for each node in constrained paths at each level of hierarchy. Therefore, the time complexity for incorporating path delay constraints is at most $O(N_{cp} \cdot N)$ at each level.

From above discussion, since $O(N \cdot (\log N + N_n + N_{cp}))$ time is required at each level, the time complexity of

Step 7. Covering process

(1) Let $t(\in N_{cand})$ be an uncovered node with the largest level [3].

(2) Generate $G_R(t)$ for t. Let $v_1 = t$.

(3) In $G_R(t)$, augment an amount of a flow from s to v_1 and compute a cut. If the size of the cut is four (i.e. the number of inputs for an LUT), go to (6).

(4) For u_1, u_2 $(\in G_R(t))$ associated with $u \in N_s \cap N_{G(t)}$,

$$N^c_{G_R(t)} = \{u_1 | u_1 \in N^s_{G_R(t)} \wedge u_2 \in N^t_{G_R(t)}\}$$

If $N^c_{G_R(t)} = \emptyset$, go to (6).

(5) Let v be a node in a path p with the tightest path delay constraint satisfying the following equation.

$$v \in N_{cand} \ \wedge \ N_p(v \rightarrow t) \subseteq N_{G(t)} \wedge$$
$$v_1 \ (\in G_R(t)) \text{ associated with } v \text{ s.t. } v_1 \in N^c_{G_R(t)}$$

For $v_1 \in N^c_{G_R(t)}$ associated with v, go to (3). If there exist no such nodes, select a node whose label is closest to the range between l_{min} and l_{max} and go to (3).

(6) For an obtained cut, compute $cover(t)$.

(7) Attempt to exclude non-candidate nodes for technology mapping from $cover(t)$.

(8) Replace $cover(t)$ with a logic-block t and execute replication. Give labels to generated blocks and nodes as in Step 5 (Section III-D) and go to (1).

Fig. 11. Covering process for reducing logic-blocks.

Maple-opt is $O(N \cdot (\log N + N_n + N_{cp}) \cdot \log L)$.

The space complexity of Maple-opt is clearly $O(N_{cp} \cdot N)$.

IV. Experimental Results

Maple-opt has been implemented on Sun Sparc Station 2 (28.5 MIPS) in C language and applied to MCNC benchmark circuits in Table I. We have compared three algorithms; Maple [12], a simultaneous placement and global routing algorithm for FPGAs [11], and the proposed algorithm, Maple-opt.

(1) Maple: First, we apply a decomposition command `xl_split` in mis-pga(new) [7] for each benchmark circuit and obtain a feasible Boolean network as input for Maple. Then, Maple is executed. In this case, path delay constraints are not treated.

(2) A simultaneous placement and global routing algorithm: First, we use mis-pga(new) for each benchmark circuit and obtain a net-list including logic-blocks. Then, placement and global routing [11] are executed. In this case, path delay constraints are treated in a layout level.

(3) Maple-opt: For the same input circuits as (1), Maple-opt is executed.

[3]First, the level of each node with no incoming edges is set to be 0. The level of other nodes is the largest level of the fanin nodes plus one.

TABLE I
Benchmark Circuits.

circuit	#primary inputs	#primary outputs	#slots
alu2	10	6	16×16
alupla	25	5	9×9
bw	5	28	11×11
duke2	22	29	16×16
f51m	8	8	9×9
misex1	8	7	7×7
misex3	14	14	17×17
misex3c	14	14	17×17
rd73	7	3	6×6
rd84	8	4	8×8
term1	34	10	12×12
vg2	25	8	10×10

The number of slots of the FPGA model in (1) – (3) is the same as that of [12] (Table I). Delay per logic-block L_d and delay per switch-block S_d are set to be three and one, respectively [6],[13]. Each path delay constraint d_{max} is defined as 85% of the maximum delay obtained by executing the algorithm (1), i.e. Maple, based on [3],[5],[8]. Constraints are given to paths with the maximum logic level in each input circuit. In (2), since we give constraints at the level of net-lists including logic-blocks, the number of paths with constraints of (2) is different from that of (1) or (3).

Tables II – IV show experimental results for path delay constraints and channel density. From those tables, Maple-opt can solve violations of path delay constraints except for two circuits. In case with consideration of path delay only in a layout level [11] or with no consideration of it [12], path delay cannot be maintained within given constraint values.

Channel density (the maximum number of tracks per channel for global routing) by Maple-opt increases by at most one track per channel compared with Maple (Table II). It is an average of 20% less than that of the case in which technology mapping and layout are executed separately (Table III). Maple-opt spends approximately 2.8 times more in CPU time compared with Maple.

From the experimental results, Maple-opt can reduce path delay to 85% by means of incorporating path delay constraints into Maple. Since delay reduction by 15% is greater or equal to the results in conventional performance-driven CAD [3],[5],[8], Maple-opt is efficient and effective in design of FPGAs where delay reduction is necessary. Moreover, since Maple-opt inherits the basic algorithm of Maple, channel density obtained by Maple-opt remains small by making good use of unused logic-

TABLE II
EXPERIMENTAL RESULTS (Maple[12]).

circuit	$\#cp$	d_{max}	$\#v$	max(d)	*dens.*	$\#lb$	$\#wl$	CPU time [s]
alu2	4	177	4	209	13	192	1740	5.7 + 3.58
alupla	12	48	12	57	7	78	386	0.8 + 0.82
bw	18	122	5	145	8	105	602	1.6 + 1.12
duke2	4	87	4	103	12	235	2264	4.7 + 4.90
f51m	30	101	14	119	7	63	323	1.3 + 0.81
misex1	4	24	4	28	4	34	79	0.4 + 0.28
misex3	16	206	7	236	11	236	2145	6.2 + 4.14
misex3c	3	218	3	257	11	231	2101	6.0 + 4.86
rd73	20	31	10	37	4	23	60	0.6 + 0.16
rd84	3	38	3	45	6	41	156	0.7 + 0.38
term1	25	82	10	97	9	136	838	1.9 + 1.54
vg2	4	29	4	35	5	69	297	0.9 + 0.48
total	143	1149	80	1368	97	1443	10991	30.8 + 23.07

$\#cp$: number of constrained paths, d_{max}: constraint value given to constrained paths, $\#v$: number of violated paths for constraints, max(d): maximum path delay among constrained paths, *dens.*: channel density, i.e. maximum number of used tracks per channel, $\#lb$: number of generated logic-blocks, $\#wl$: total wire length, i.e. total number of tracks occupied by nets, CPU time: time for `xl_split` + Maple

TABLE III
EXPERIMENTAL RESULTS (mis-pga(new)[7] + a simultaneous placement and global routing algorithm[11]).

circuit	$\#cp$	d_{max}	$\#v$	max(d)	*dens.*	$\#lb$	$\#wl$	CPU time [s]
alu2	8	177	8	209	16	152	1562	431.4 + 3.95
alupla	32	48	22	59	7	68	385	50.5 + 1.44
bw	36	122	2	124	10	91	665	25.1 + 4.03
duke2	9	87	9	105	17	192	1924	341.2 + 3.63
f51m	60	101	60	122	7	51	251	33.9 + 4.32
misex1	3	24	3	32	4	28	80	7.9 + 0.08
misex3	4	206	0	185	16	190	1985	430.9 + 3.80
misex3c	6	218	6	286	15	189	1940	248.9 + 5.55
rd73	16	31	0	31	5	23	83	36.9 + 0.20
rd84	86	38	0	37	6	33	165	98.1 + 1.41
term1	6	82	6	100	12	110	839	72.6 + 1.45
vg2	12	29	12	49	6	59	358	10.9 + 0.91
total	278	1149	128	1339	121	1186	10237	1791.0 + 30.77

CPU time: time for mis-pga(new) + time for a simultaneous placement and global routing algorithm

TABLE IV
EXPERIMENTAL RESULTS (Maple-opt).

circuit	$\#cp$	d_{max}	$\#v$	max(d)	*dens.*	$\#lb$	$\#wl$	CPU time [s]
alu2	4	177	0	171	13	190	1945	5.7 + 10.92
alupla	12	48	1	52	7	75	382	0.8 + 1.17
bw	18	122	0	122	9	102	633	1.6 + 2.97
duke2	4	87	0	87	12	230	2144	4.7 + 5.65
f51m	30	101	1	105	8	61	319	1.3 + 3.07
misex1	4	24	0	24	4	37	90	0.4 + 0.35
misex3	16	206	0	200	11	231	2282	6.2 + 18.22
misex3c	3	218	0	218	11	235	2204	6.0 + 15.48
rd73	20	31	0	31	4	23	65	0.6 + 0.20
rd84	3	38	0	38	6	39	162	0.7 + 0.43
term1	25	82	0	82	10	137	888	1.9 + 4.03
vg2	4	29	0	29	5	67	334	0.9 + 0.71
total	143	1149	2	1159	100	1427	11448	30.8 + 63.20

CPU time: time for `xl_split` + time for Maple-opt

blocks. Maple-opt can realize given circuits inside given FPGA chips and satisfy performance requirement.

V. Conclusion

We have proposed Maple-opt, a simultaneous technology mapping, placement, and global routing algorithm for FPGAs incorporating path delay constraints. Experimental results demonstrate that it reduces path delay sufficiently.

Acknowledgment

This work was supported by Kanagawa Academy of Science and Technology.

References

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

[2] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field-programmable gate arrays*, Kluwer Academic Publishers, 1992.

[3] 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," *Proc. ICCAD-93*, pp. 123-127, 1993.

[4] J. Cong and Y. Ding, "An optimal technology mapping algorithm of delay optimization in lookup-table based FPGA designs," *Proc. ICCAD-92*, pp. 48-53, 1992.

[5] A. E. Dunlop, V. D. Agrawal, D. N. Deutsch, M. F. Jukl, P. Kozak, and M. Wiesel, "Chip layout optimization using critical path weighting," *Proc. 21st DA Conf.*, pp. 133-136, 1984.

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

[7] R. Murgai, N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Improved logic synthesis algorithms for table look up architectures," *Proc. ICCAD-91*, pp. 564-567, 1991.

[8] Y. Ogawa, T. Ishii, Y. Shiraishi, H. Terai, T. Kozawa, K. Yuyama, and K. Chiba, "Efficient placement algorithms optimizing delay for high-speed ECL masterslice LSI's," *Proc. 23rd DA Conf.*, pp. 404-410, 1986.

[9] R. E. Tarjan, *Data structures and network algorithms*, Society for Industrial and Applied Mathematics, 1983.

[10] M. Terai, K. Takahashi, and K. Sato, "A new min-cut placement algorithm for timing assurance layout design meeting net length constraint, " *Proc. 27th DA Conf.*, pp. 96-102, 1990.

[11] N. Togawa, M. Sato, and T. Ohtsuki, "A simultaneous placement and global routing algorithm for FPGAs," *Proc. ISCAS'94*, pp. 483-486, 1994.

[12] N. Togawa, M. Sato, and T. Ohtsuki, "Maple: A simultaneous technology mapping, placement, and global routing algorithm for Field-Programmable Gate Arrays," *Proc. ICCAD-94*, pp. 156-163, 1994.

[13] XILINX, *The programmable logic data book*, 1993.