IEICE TRANS. FUNDAMENTALS, VOL.E83–A, NO.12 DECEMBER 2000

PAPER Special Section on VLSI Design and CAD Algorithms

Clock Schedule Design for Minimum Realization Cost

Tomoyuki YODA[†*], *Nonmember and* **Atsushi TAKAHASHI**[†], *Regular Member*

SUMMARY A semi-synchronous circuit is a circuit in which the clock is assumed to be distributed periodically to each individual register, though not necessarily to all registers simultaneously. In this paper, we propose an algorithm to achieve the target clock period by modifying a given target clock schedule as small as possible, where the realization cost of the target clock schedule is assumed to be minimum. The proposed algorithm iteratively improves a feasible clock schedule. The algorithm finds a set of registers that can reduce the cost by changing their clock timings with same amount, and changes the clock timing with optimal amount. Experiments show that the algorithm achieves the target clock period with fewer modifications.

key words: *semi-synchronous circuit, clock schedule, clock tree*

1. Introduction

A *semi-synchronous circuit* is a circuit in which the clock is assumed to be distributed periodically to each individual register, though not necessarily to all registers simultaneously. Among various objectives in the synthesis of high-performance circuits, the clock period minimization is the primal subject. For a given circuit with fixed signal propagation delays between registers, there exists a lower bound of the clock period in semi-synchronous framework which is usually smaller than the maximum signal delay between registers. This lower bound is achieved if the clock is distributed to each register at proper timing [4].

In designing a synchronous circuit, a target clock period is usually set. Even if the target clock period can be theoretically achieved for a given circuit in semi-synchronous framework, a clock schedule must be realized in layout synthesis. It is shown that an arbitrary clock schedule can be realized by constructing a clock tree [3]. However, there are many clock schedules that achieve the target clock period, each of which has its own realization cost. Therefore, it is important to get a clock schedule that achieves the target clock period with a lower realization cost.

It is not easy to give a concrete definition of the realization cost of a clock schedule. The cost of a clock tree that realizes a clock schedule will consist of the wire length, the power consumption, and so on. However, it is difficult to find an optimal clock tree that realizes the clock schedule. Therefore, we adopt an assumption that the cost of a clock schedule is proportional to the difference from the *target clock schedule*. The target clock schedule is defined by a hypothetical optimal clock tree. The realization cost of a clock tree that realizes the clock schedule would be small if the difference between the clock schedule and the target clock schedule is small. If the target clock schedule achieves the target clock period, we could get a desired circuit by constructing the assumed optimal clock tree, if not, we should modify the target clock schedule to achieve the target clock period. The problem is to find a clock schedule that achieves the target clock period and that minimizes the difference from the target clock schedule.

In the clock driven layout methodology [5], a circuit layout is obtained by assuming an optimal clock tree. Then, a clock tree is constructed by using information of the layout. In this case, the target clock schedule will be the clock schedule defined by the assumed clock tree. The clock scheduling method to get the clock schedule in which the clock timing of each register is near to zero is proposed in [1]. The modification method of the clock tree to enhance the reliability of the circuit is proposed in [6].

We propose an algorithm to achieve the target clock period by modifying a given target schedule as small as possible. Experiments show that the algorithm achieves the target clock period with fewer modifications.

The rest of this paper is organized as follows. In Sect. 2, we give a basic definitions. The target scheduling algorithm is proposed in Sect. 3. The experimental results are presented in Sect. 4. Section 5 is the conclusion.

2. Preliminaries

In this paper, we consider a circuit with a single clock consisting of registers and combinatorial circuits between them. The *clock timing* $s(v)$ of register v is the difference in clock arrival time between v and an arbitrary chosen (perhaps hypothetical) reference register. The set of clock timings is called a *clock schedule*. Let $d_{\max}(u, v)$ ($d_{\min}(u, v)$) be the maximum (minimum) propagation delay from register u to register v along a combinatorial circuit.

We assume the framework that a circuit works cor-

Manuscript received March 31, 2000.

[†]The authors are with the Department of Communications and Integrated Systems, Tokyo Institute of Technology, Tokyo, 152-8552 Japan.

[*]Presently, with Toshiba Corporation.

rectly with clock period T if the following two types of constraints are satisfied for every register pair with signal propagation [2].

No-Double-Clocking Constraints

$s(v) - s(u) \leq d_{\min}(u, v)$

No-Zero-Clocking Constraints

$s(u) - s(v) \leq T - d_{\max}(u, v)$

The *constraint graph* $G(V, E)$ is defined as follows: a vertex $v \in V$ corresponds to a register, and a directed edge $(u, v) \in E$ corresponds to either type of constraints. An edge (u, v) called D-edge (Z-edge) corresponds to the no-double (no-zero) clocking constraint, and the weight $w(u, v)$ of the edge is $d_{\min}(u, v)$ $(T - d_{\max}(v, u))$. For a clock schedule s, an edge (u, v) is said to be *legal* if $s(v) - s(u) \leq w(u, v)$, *illegal* otherwise. The slack of an edge (u, v) is

$$\Delta(u, v) = w(u, v) - (s(v) - s(u)).$$

If the slack of an edge is zero, the edge is said to be *critical*. A clock schedule is called *feasible* in clock period T if there is no illegal edge in the constraint graph. The circuit works with the feasible clock schedule when the clock period is T. There exists a feasible clock schedule if the constraint graph contains no negative weight directed cycle.

For example, the constraint graph of the circuit shown in Fig. 1 is shown in Fig. 2. In Fig. 2, the number in each vertex corresponds to the clock timing defined by the clock tree in the circuit shown in Fig. 1. If $T < 9$, edges (r_4, r_1) and (r_1, r_2) are illegal, and if $T \geq 9$, they are legal. They are critical when $T = 9$. There is no illegal edge if $T \geq 9$. So, the minimum clock period of this circuit is 9. The clock period of this circuit is at least 9 even if the clock schedule is modified. This can be easily verified [4]. However there are many clock schedules that achieve the minimum clock period 9.

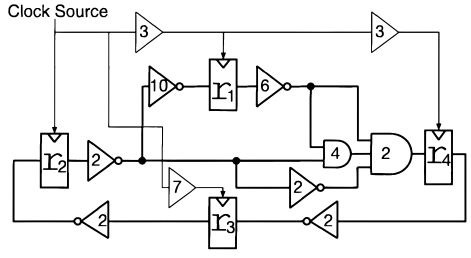

Fig. 1 A semi-synchronous circuit C.

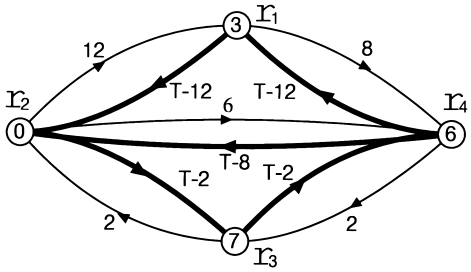

Fig. 2 Constraint graph of C in Fig. 1.

3. Target Clock Scheduling

We propose an algorithm to obtain a feasible clock schedule that achieves the target clock period T and that minimizes the difference from the given target schedule s_o. It is trivial if s_o is feasible. Thus we assume that s_o is infeasible and there exists a feasible clock schedule. Usually, there are various feasible clock schedules of various realization costs. The algorithm selects a clock schedule with minimum realization cost.

The target scheduling problem
Input: constraint graph $G(V, E)$, target schedule s_o
Output: clock schedule s such that $s(v) - s(u) \leq w(u, v)$ for $\forall(u, v) \in E$
Objective: minimize $\sum_{v \in V} |s(v) - s_o(v)|$

For the given target schedule s_o, we replace $w(u, v)$ with $w(u, v) - s_o(v) + s_o(u)$ for all edges (u, v) in E. Then we can regard the target clock timing of vertex v as zero. In the following, without loss of generality, we assume that $s_o(v) = 0$ for all v in V.

The proposed algorithm iteratively improves a feasible clock schedule. A feasible initial clock schedule is obtained by the algorithm shown in [4]. In each iteration, the algorithm finds a set of registers that can reduce the cost by changing their clock timings with same amount, and changes the clock timing with optimal amount. This operation is repeated until a minimum cost schedule is obtained. An outline of the algorithm is shown in Fig. 3.

The detailed algorithm to get an optimal amount α_{opt} is explained in Sect. 3.1. In Sect. 3.2, the detailed algorithm to find a set of registers which reduce the cost by changing their clock timings with same amount is explained.

3.1 Optimal Amount of Clock Timing Adjustment

Let s be a feasible clock schedule, and $R \subseteq V$ be a

Step 1 Find a feasible initial clock schedule.
Step 2 Find a set R of registers which can reduce the cost by changing the clock timing of each register in R with the same amount.
Step 3 Determine the optimal amount α_{opt} such that the cost is minimized when the clock timing $s(v)$ of register v in R is changed to $s(v) - \alpha_{opt}$.
Step 4 Repeat step 2 and step 3 until there is no such set of registers.

Fig. 3 Outline of clock scheduling algorithm.

set of registers. Let s' be a clock schedule such that $s'(v) = s(v) - \alpha$ if v in R, $s'(v) = s(v)$ otherwise.

The edge (u, v) is legal in s' if both u and v are in R or neither u and v are in R. In case that $u \in R$ and $v \notin R$, (u, v) is legal if $w(u, v) - s(v) + (s(u) - \alpha) \geq 0$, that is, $\alpha \leq w(u, v) - s(v) + s(u) = \Delta(u, v)$. Similarly, in case that $u \notin R$ and $v \in R$, (u, v) is legal if $w(u, v) - (s(v) - \alpha) + s(u) \geq 0$, that is, $\alpha \geq -(w(u, v) - s(v) + s(u)) = -\Delta(u, v)$. Thus, s' is feasible if and only if

$$L(R) \leq \alpha \leq U(R), \tag{1}$$

where $L(R) = \max_{u \notin R, v \in R}(-\Delta(u, v))$ and $U(R) = \min_{u \in R, v \notin R} \Delta(u, v)$.

The cost of s' is minimum when the number of registers in R with positive clock timing is equal to that in R with negative clock timing. If such schedule is infeasible, then the cost of s' is minimum when the difference in the number between registers with positive clock timing and that with negative clock timing is minimum.

Let β be zero, if the number n of registers in R is even and $\lfloor \frac{n+1}{2} \rfloor$-th largest clock timing in R is non-negative in s and $\lceil \frac{n+1}{2} \rceil$-th in R is non-positive in s. Otherwise let β be the $\lfloor \frac{n+1}{2} \rfloor$-th largest clock timing of registers in R in s.

Lemma 1: The optimal α, say α_{opt}, that minimizes the cost of s' is β if $L(R) \leq \beta \leq U(R)$, $L(R)$ if $\beta < L(R)$, and $U(R)$ if $U(R) < \beta$.

If α_{opt} is zero, the set of registers cannot reduce the cost. If α_{opt} is positive or negative, then the cost can be reduced. α_{opt} is positive if and only if both $U(R)$ and β are positive, that is, there is no critical out-edge from R in s, and the number of registers in R with positive clock timing in s is larger than that in R with non-positive clock timing in s. Similar discussion is possible when α_{opt} is negative.

3.2 Cost Reduceable Register Sets

The problem is to find a set R of registers with non-zero α_{opt}. In the following, we discuss the problem that finds a set R of registers with positive α_{opt}. Similar discussion is possible in the case of negative α_{opt}.

Input: The constraint graph $G(V, E)$, and clock schedule s.
Question: Is there a set R of registers which satisfies the following conditions.

- There is no critical out-edge from R in s.
- The number of registers in R with positive clock timing in s is larger than that with non-positive clock timing in s.

Let $G_c(V, E_c)$ be the critical graph obtained from constraint graph $G(V, E)$ by deleting non-critical edges. A vertex is labeled "+" if the clock timing is positive in s, labeled "−" otherwise.

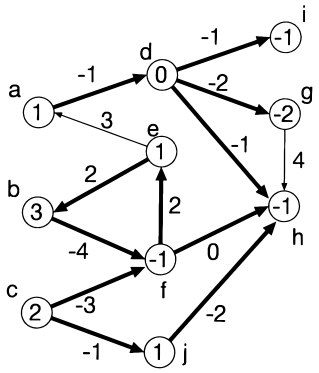

Fig. 4 Constraint graph and critical edges.

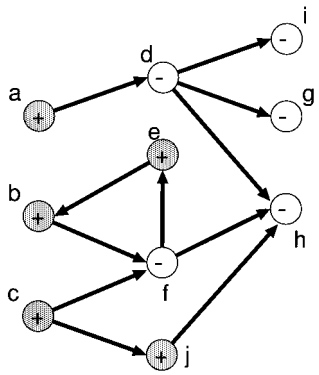

Fig. 5 Critical graph from constraint graph.

The problem is formulated as follows.

Input: The critical graph $G_c(V, E_c)$
Question: Is there a set R of vertices which satisfies following conditions.

- There is no out-edge from R.
- The number of "+" vertices in R is larger than that of "−" vertices in R.

An illustrative constraint graph is shown in Fig. 4. Critical edges are written in bold. The corresponding critical graph is shown in Fig. 5.

The algorithm which solves the problem is shown in Fig. 6.

We explain the algorithm using the critical graph shown in Fig. 5. The graph G_H obtained in step 1 and step 2 is shown in Fig. 7. The edges in the maximum matching obtained in step 3 are written in bold in Fig. 7. In Fig. 8, the graph G_H^* is shown. Vertices a, b, c, f, g, and h are matched vertices, and others are unmatched vertices. Because e is unmatched "+" vertex, e is selected as v_s in step 5. Reachable vertices from e are shown in Fig. 8. Because the vertex j is reachable from vertex $c \in R'$ in graph G_C, j is added to R in step 6. Thus, $R = \{b, c, e, f, h, j\}$ is obtained.

We prove that the algorithm finds a set R of vertices which satisfies the conditions as follows.

Step 1 Let $G'_H(V, E'_H)$ be the transitive graph of $G_C(V, E_C)$, that is, there is an edge (u, v) in E'_H if there is a path from u to v in G_C.

Step 2 Let $G_H(V, E_H)$ be the graph obtained from $G'_H(V, E'_H)$ by deleting all edges in E'_H except the edges connecting from "+" vertex to "−" vertex.

Step 3 Obtain a maximum matching of the underlying graph of G_H, and let G^*_H be the graph obtained from G_H by changing the direction of all matched edges.

Step 4 If all "+" vertices is matched, then output "No" and terminate.

Step 5 Select an unmatched "+" vertex v_s, and let R' be the set of vertices reachable from v_s in G^*_H.

Step 6 Output the set R of vertices obtained from R' by adding a vertex reachable from R' in $G_C(V, E_C)$.

Fig. 6 Cost reduceable set finding algorithm.

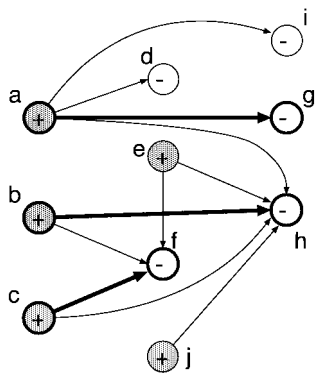

Fig. 7 Maximum matching of graph G_H.

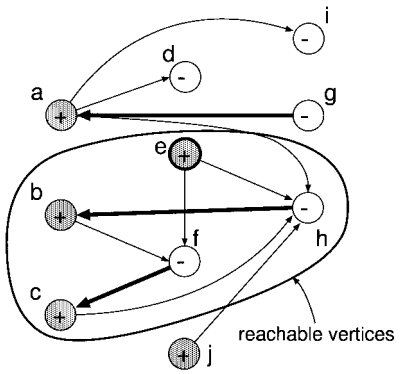

Fig. 8 Graph G^*_H and reachable vertices from e.

1. If the algorithm outputs "No," no set of vertices satisfies the conditions.
2. No edge (u, v) with $u \in R$ and $v \notin R$ exists in G_C.
3. A set R of vertices which is given by the algorithm has more "+" vertices than "−" vertices.

Theorem 1: If all "+" vertices are matched in step 3, then there is no set of vertices that satisfies the two conditions.

Proof: Assume that there exists a set A of vertices that satisfies the two conditions. For each "+" vertex u in A, there exists a "−" vertex v which is matched to u. Then there exists a path from u to v in the critical graph G_C. By the former condition, v is contained in A, for otherwise, there exists an edge on the path such that the edge is out-edge from A in G_C. Therefore, the number of "−" vertices is at least that of "+" vertices. This contradicts the latter condition. □

Theorem 2: No out-edge from R exists in G_C.

Proof: Assume that there exists an edge from $u \in R$ to $v \notin R$ in G_C. In step 6, v is added to R, since v is reachable from u in G_C. This contradicts the assumption. □

Lemma 2: Except v_s, no unmatched vertex exists in R' obtained in step 5.

Proof: An unmatched "+" vertex has no in-edge in G^*_H since the direction of edge is from "+" vertex to "−" vertex except matched edges. Thus, every unmatched "+" vertex except v_s is not reachable from v_s in G^*_H, and not contained in R'.

If there exists a path from v_s to an unmatched "−" vertex in G^*_H, then the path is an augmenting path of the matching. This contradicts the fact that the maximum matching is obtained in step 3. □

Lemma 3: The number of "+" vertices in R' is larger than the number of "−" vertices in R'.

Proof: If a matched "+" vertex u is contained in R', then the corresponding "−" vertex v is contained in R', since (v, u) is the only in-edge of u in G^*_H. Thus, the number of "+" matched vertices is equal to that of "−" matched vertices in R'. From Lemma 2, R' contains one unmatched "+" vertex. So, the number of "+" vertices is just one larger than the number of "−" vertices. □

Lemma 4: All vertices added to R' in step 6 are "+" vertices.

Proof: For each "−" vertex u in R', there exists an edge (w, u) in G_H such that w is "+" vertex in R', since there is no edge connecting "−" vertices in G_H. That is, for each u, there exists a "+" vertex w in R' such that u is reachable from w in G_C. Assume that "−" vertex $v \notin R'$ is added to R' in step 6. Then v is reachable from a vertex in R' in G_C. That is, v is reachable from "+" vertex w in R' in G_C. Therefore, there exists an edge (w, v) in G_H. This contradicts that v is not in R'. □

From Lemmas 3 and 4, we have the following theorem.

Theorem 3: The number of "+" vertices in R is larger than that of "−" vertices in R.

Thus, every claim is proved and the algorithm finds cost reduceable register set if it exists.

Theorem 4: The algorithm outputs the minimum cost schedule.

Proof: Let s be the clock schedule obtained by the algorithm. Assume contrary that there exists a clock schedule s' such that $\sum|s'(v)| < \sum|s(v)|$.

Let $\delta(v) = s(v) - s'(v)$. First, we show that if an edge (u, v) is critical in s, then $\delta(u) \leq \delta(v)$. Since s is feasible and (u, v) is critical, we have $s(v) - s(u) = w(u, v)$. Since s' is feasible, $s'(v) - s'(u) \leq w(u, v)$. Since $(s(v) - s'(v)) - (s(u) - s'(u)) \geq 0$, we have $\delta(u) \leq \delta(v)$.

We partition the vertex set V into $V_+ = \{v|\delta(v) > 0\}$, $V_- = \{v|\delta(v) < 0\}$, and $V_0 = \{v|\delta(v) = 0\}$. Further we partition the vertex set V_+ into V_1, V_2, ..., V_m as follows: for any vertex v and u in V_i $(1 \leq i \leq m)$, $\delta(v) = \delta(u) = \delta_i$; for any vertex $v \in V_i$ and $u \in V_j$ $(1 \leq i < j \leq m)$, $\delta(v) > \delta(u)$.

Let $f_i(x) = \sum_{v \in V_1 \cup V_2 \cup \cdots \cup V_i} |s(v) - x|$. Since there is no cost reduceable set for s, $f_i(x) \geq f_i(x')$ for any $x > x' \geq 0$. Note that there is no critical out-edge from $\{V_1, V_2, \ldots, V_i\}$, and $f_i(x)$ is a convex function. Then $\sum_{v \in V_+} |s'(v)| = f_1(\delta_1) - f_1(\delta_2) + f_2(\delta_2) - \cdots - f_{m-1}(\delta_m) + f_m(\delta_m) \geq f_m(0) = \sum_{v \in V_+} |s(v)|$. Similarly, we have $\sum_{v \in V_-} |s'(v)| \geq \sum_{v \in V_-} |s(v)|$. Therefore, $\sum_{v \in V} |s'(v)| \geq \sum_{v \in V} |s(v)|$, and this contradicts the assumption. □

In example shown in Fig. 4, $\{b, c, e, f, h, j\}$ is selected as R, and optimal amount α_{opt} is 1. After changing clock timings of registers in R, there exists no cost reduceable set. The schedule obtained by the algorithm is shown in Fig. 9.

4. Experimental Results

The proposed algorithm and an algorithm using linear programming are implemented on PentiumII 450MHz (Free BSD). They are applied to benchmark circuits in LGSynth93. The experimental results of the target scheduling algorithm are shown in Table 1. We assume that the signal delay between registers is the number of gates along a path in the combinatorial circuits between them. The target clock timing of each register is zero. In Table 1, the number of registers is shown in column #Reg, the number of register pairs with signal propagation in column #Path, the maximum delay in a circuit in column Max D, the minimum clock period in semi-synchronous framework in column Min CP which is the target clock period, the number of registers scheduled other than the target clock timing in column #Other, the sum of the difference between the resultant clock timing and the target clock timing in column Cost, the computational time of the proposed algorithm in column Time, and the computational time using the linear programming in LP. Although the clock schedule obtained by our algorithm is different from that by linear programming, the costs of them are same.

5. Conclusion and Future Works

In semi-synchronous framework, we proposed the algorithm to minimize the difference from the target clock schedule. In experiments, the minimum clock period is achieved with fewer difference from the target clock timing. Our algorithm is faster than the algorithm using linear programming. As future works, we should discuss the validity of cost function and improve it.

Acknowledgments

The authors are grateful to Professor Yoji KAJITANI of Tokyo Institute of Technology for his helpful comments and supports. This work is part of a project of CAD21 at Tokyo Institute of Technology.

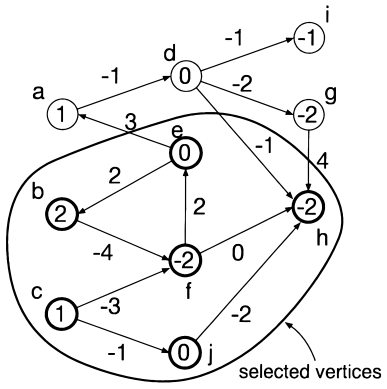

Fig. 9 Solution given by the scheduling algorithm.

Table 1 Results of the target scheduling algorithm.

Model	#Reg	#Path	Max D	Min CP	#Other	Cost	Time [sec]	LP [sec]
s1423	75	1766	66	57.00	6	21.00	0.093	0.141
s5378	164	1185	29	22.34	16	22.53	0.087	0.140
s9234.1	135	1947	55	51.00	16	30.00	0.107	0.182
s38417	1458	31446	65	45.00	63	149.00	2.393	13.080
s38584.1	1427	16219	67	48.00	12	48.00	1.361	2.095

References

[1] R.B. Deokar and S.S. Sapatnekar, "A graph-theoretic approach to clock skew optimization," Proc. IEEE International Symposium on Circuits and Systems, vol.1, pp.407–410, 1994.
[2] J.P. Fishburn, "Clock skew optimization," IEEE Trans. Comput., vol.39, no.7, pp.945–951, 1990.
[3] A. Takahashi, K. Inoue, and Y. Kajitani, "Clock-tree routing realizing a clock-schedule for semi-synchronous circuits," Proc. IEEE/ACM International Conference on Computer Aided Design, pp.260–265, 1997.
[4] A. Takahashi and Y. Kajitani, "Performance and reliability driven clock scheduling of sequential logic circuits," Proc. Asia and South Pacific Design Automation Conference, pp.37–42, 1997.
[5] A. Takahashi, W. Takahashi, and Y. Kajitani, "Clock-routing driven layout methodology for semi-synchronous circuit design," Proc. IEEE/ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU'97), pp.63–66, 1997.
[6] J.G. Xi and W.W. M. Dai, "Useful-skew clock routing with gate sizing for low power design," Proc. 33rd Design Automation Conference, pp.383–388, 1996.

Tomoyuki Yoda received his B.E. and M.E. degrees in electrical and electronic engineering from the Tokyo Institute of Technology, Tokyo, Japan, in 1998 and 2000, respectively. He currently works at Toshiba Corporation. His research interests are in VLSI design automation and combinational algorithms.

Atsushi Takahashi received his B.E., M.E., and D.E. degrees in electrical and electronic engineering from the Tokyo Institute of Technology, Tokyo, Japan, in 1989, 1991, and 1996, respectively. He had been with the Tokyo Institute of Technology as a research associate from 1991 to 1997 and has been an associate professor since 1997. He has been an associate professor in Department of Communications and Integrated Systems since 2000. His research interests are in VLSI layout design and combinational algorithms. He is a member of the Information Processing Society of Japan and IEEE.