T2R2 東京工業大学リサーチリポジトリ
Tokyo Tech Research Repository

論文 / 著書情報
Article / Book Information

Title	Minimization of Delay Insertion in Clock Period Improvement in General-Synchronous Framework
Authors	Yukihide Kohira, Shuhei Tani, Atsushi Takahashi
Citation	IEICE Trans. Fundamentals, Vol. E92-A, No. 4, pp. 1106-1114
Pub. date	2009, 4
URL	http://search.ieice.org/
Copyright	(c) 2009 Institute of Electronics, Information and Communication Engineers

Powered by T2R2 (Tokyo Institute Research Repository)

IEICE TRANS. FUNDAMENTALS, VOL.E92–A, NO.4 APRIL 2009

PAPER Special Section on Advanced Technologies Emerging Mainly from the 21st Workshop on Circuits and Systems in Karuizawa

Minimization of Delay Insertion in Clock Period Improvement in General-Synchronous Framework*

Yukihide KOHIRA[†a)], *Member*, **Shuhei TANI**[†], *Nonmember*, *and* **Atsushi TAKAHASHI**[†], *Member*

SUMMARY In general-synchronous framework, in which the clock is distributed periodically to each register but not necessarily simultaneously, the circuit performance such as the clock period is expected to be improved by delay insertion. However, if the amount of inserted delays is too much, then the circuit is changed too much and the circuit performance might not be improved. In this paper, we propose an efficient delay insertion method that minimizes the amount of inserted delays in the clock period improvement in general-synchronous framework. In the proposed method, the amount of inserted delays is minimized by using an appropriate clock schedule and by inserting delays into appropriate places in the circuit. Experiments show that the proposed method can obtain optimum solutions in short time in many cases.

key words: *delay insertion, clock scheduling, general-synchronous framework*

1. Introduction

The semiconductor manufacturing process technology has improved the scale, speed, and power consumption of LSI circuits. However, increasing the ratio of the routing delay in the propagation delay bounds the amount of improvements in complete-synchronous framework (c-frame) in which the simultaneous clock distribution to every register is assumed. The increases of the size and power consumption of a clock distribution circuit have become serious issues in c-frame. While, general-synchronous framework (g-frame) [2]–[4], in which the clock is assumed to be distributed periodically to each individual register though not necessarily to all the registers simultaneously, is expected to give an essential solution. By using g-frame, the quality of circuit such as the clock frequency, clock distribution circuit size, power consumption, and etc. are expected to be improved. The efforts toward improvements of qualities in g-frame are summarized in [5].

Since the clock period might not be reduced in g-frame even if the maximum delay is reduced, the effort in c-frame might degrade the circuit performance in g-frame. Therefore, the optimization of circuit synthesis that takes g-frame into account must be investigated. In this paper, we focus on delay insertion methods [6]–[9]. Even though we do not consider the detailed delay insertion methods, the delay insertion will be realized by replacing a large module with a small module synthesized under looser delay constraints, by using smaller transistors and narrower wires, and by deleting buffers from long interconnects, as well as by inserting buffers to short interconnects.

In [6], a delay insertion method that minimizes the clock period was proposed. Since the amount of an inserted delay is iteratively determined by searching the whole circuit, the method takes too much computation time. In [7], a fast delay insertion method that minimizes the clock period was proposed. Although it is fast and the amount of inserted delays is smaller than that by the method in [6], a lot of redundant delays are still inserted. In [8] and [9], the mixed integer linear programming (MILP) formulations that minimize the clock period and that minimize the amount of inserted delays at the given clock period were proposed, respectively. Although optimum solutions are obtained by MILP, the methods based on MILP cannot be applied to large circuits since the required memory size and computation time of MILP are large. Although the computational complexity of the problem that minimizes the amount of inserted delays in a circuit at the given clock period in g-frame is not known, we conjecture that this problem is NP-hard.

In this paper, we propose an efficient delay insertion method that minimizes the amount of inserted delays in clock period improvement in g-frame. The proposed method is based on the method in [7]. In the method in [7], a clock schedule for a target clock period that has timing violations is assumed, and delays are greedily inserted to recover the timing violations of the assumed clock schedule. The amount of inserted delays by the method in [7] is relatively large since it assumes an inappropriate clock schedule and adopts a greedy delay insertion approach. In the proposed method, a clock schedule that has fewer timing violations is assumed in order to reduce the amount of inserting delays. Furthermore, delays are inserted into appropriate places so that the timing violations are recovered by fewer inserted delays. Experiments show that the proposed method can obtain optimum solutions in short time in many cases.

The rest of the paper is organized as follows. Section 2 provides some definitions. Section 3 discusses existing delay insertion methods. Section 4 describes our proposed clock scheduling method and delay insertion method. In Sect. 5, the experimental results are given. Section 6 concludes with a summary and future works.

Manuscript received June 25, 2008.
Manuscript revised October 31, 2008.

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

*The preliminary vesion was presented at [1].

a) E-mail: kohira@lab.ss.titech.ac.jp

DOI: 10.1587/transfun.E92.A.1106

Copyright © 2009 The Institute of Electronics, Information and Communication Engineers

2. Preliminaries

In this paper, we consider a circuit consisting of registers, gates, and wires connecting registers and gates. We refer to registers, gates, and wires as elements. A circuit is represented by the graph $G = (V_g, E_g)$, where V_g is the vertex set corresponding to elements in the circuit and E_g is the directed edge set corresponding to signal propagations in the circuit. In this paper, we assume that the maximum delay of each element is equal to its minimum delay. Let $d(v)$ be the weight of $v \in V_g$ which corresponds to the delay of corresponding element. Let V_r be a register set. Necessarily, the register set is a subset of V_g. An example of the circuit is shown in Fig. 1(a). In Fig. 1(a), $\{a, b, c, d\}$ is the register set, and the figure in each vertex except registers represents its weight.

In general-synchronous framework (g-frame), the clock arrival timing of a register may be different from other registers. The *clock timing* $S(r)$ of register r is defined as the difference in clock arrival time between r and an arbitrary chosen reference register. Moreover, the set of clock timing of all the registers S is called *clock schedule*.

A circuit works correctly with a clock period T if the following two types of constraints are satisfied for every register pair with signal propagations [2].

Setup (No-Zero-Clocking) Constraints

$$S(a) - S(b) \leq T - D_{\max}(a, b)$$

Hold (No-Double-Clocking) Constraints

$$S(b) - S(a) \leq D_{\min}(a, b),$$

where $D_{\max}(a, b)$ is the maximum delay and $D_{\min}(a, b)$ is the minimum delay from a register a to b (Fig. 2).

Since a clock ticks all the register simultaneously in complete-synchronous framework (c-frame), the clock period must be larger than or equal to the maximum delay between registers. On the other hand, in g-frame, circuits can work correctly with the clock period which is smaller than the maximum delay between registers, if all the register pair with signal path satisfies two types of constraints.

Let $T_S(G)$ be the minimum clock period of a circuit G in g-frame under the assumption that the clock can be inputted to each register at an arbitrary designated timing. Hereafter, we simply call $T_S(G)$ the minimum clock period of G in g-frame. $T_S(G)$ is determined by the *constraint graph* $H(G) = (V_r, E_r)$ for G, where vertex set V_r corresponds to registers in G and directed edge set E_r corresponds to two types of constraints [3], [4]. An edge in E_r from a register a to a register b with weight $D_{\min}(a, b)$, called the D-edge, corresponds to the hold constraint, and an edge from a register b to a register a with weight $T - D_{\max}(a, b)$, called the Z-edge, corresponds to the setup constraint. Let $H(G, t)$ be the constraint graph in which the clock period T of Z-edges in $H(G)$ is set to t. Let the weight of a directed cycle

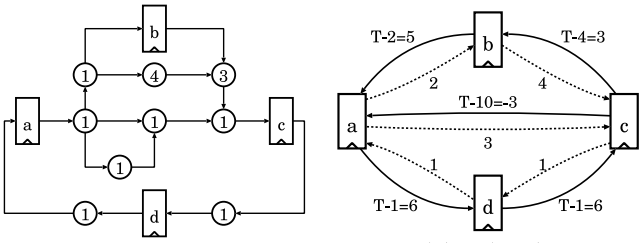


Fig. 1 A circuit G and a constraint graph $H(G, 7)$.

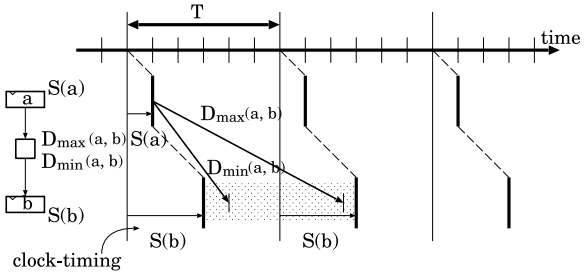


Fig. 2 Timing chart.

in $H(G, t)$ be the sum of edge weights on the directed cycle. It is known that the minimum clock period $T_S(G)$ is the minimum t such that there is no cycle with negative weight in the constraint graph $H(G, t)$.

Although the clock period of G is bounded by $T_S(G)$, a faster circuit G' in g-frame might be obtained from G by delay insertion. It is known the lower bound of the minimum clock period $T_L(G)$ of a circuit G in g-frame by delay insertion is defined as follows.

$$T_L(G) = \max_{C \in \text{cycles in } G} \frac{D(C)}{N(C)},$$

where $N(C)$ and $D(C)$ are the number of registers and and the sum of delay on a directed cycle C in G, respectively. $T_L(G)$ is the minimum t such that there is no cycle with negative weight in the constraint graph $H(G_Z, t)$ whose edge set consists of Z-edges only [7]. Note that $T_L(G) \leq T_S(G)$.

For example, the constraint graph H(G, 7) of G shown in Fig. 1(a) is shown in Fig. 1(b). Since the clock period must be larger than or equal to the maximum delay between registers in c-frame, the minimum clock period of G in c-frame $T_C(G) = 10$. Since H(G, 7) has no cycle with negative weight and the weight of cycle (a, c, a) is negative when $T < 7$, $T_S(G) = 7$. Moreover, $T_L(G) = 4$, which is determined by cycle (a, c, d, a) in the constraint graph with only Z-edges.

3. Existing Delay Insertion Method

The existing delay insertion methods can be classified into heuristic methods [6], [7] and mixed integer linear programming (MILP) formulations [8], [9]. Although optimum solutions are obtained by MILP, the methods based on MILP cannot be applied to large circuits since the required memory size and computation time of MILP are large. Although

Procedure **SchedulingZ**($H(G,t)$)
Input : constraint graph $H(G,t)$
Output : clock schedule S
Step 1 : Determine a clock schedule S that satisfies all the setup constraints.
Step 2 : return S.

Procedure **InsertDelayDFS**(G,S)
Input : circuit G, clock schedule S
Output : circuit G' after delay insertion
Step 1 : Until all the hold constraints are satisfied, repeat the following.

1. Calculate delay-slack and delay-demand.
2. Find an edge e by DFS from a register from which a hold constraint is violating, and insert a delay which equals to $\min\{slack(e), demand(e)\}(>0)$ to e.

Step 2 : return G.

Input : circuit G, target clock period t
Output : circuit G' after delay insertion
Step 1 : $S_{init} :=$ SchedulingZ($H(G,t)$)
Step 2 : $G' :=$ InsertDelayDFS(G, S_{init})
Step 3 : return G'.

Fig. 3 The method in [7] (S_{init}+DFS).

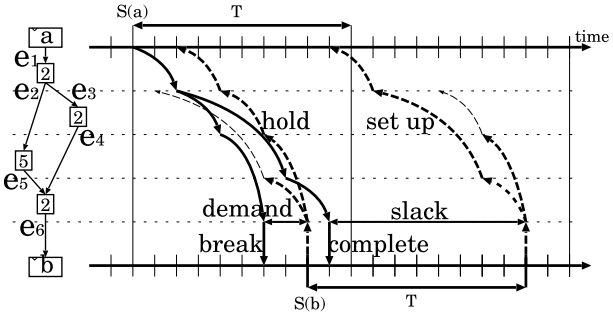
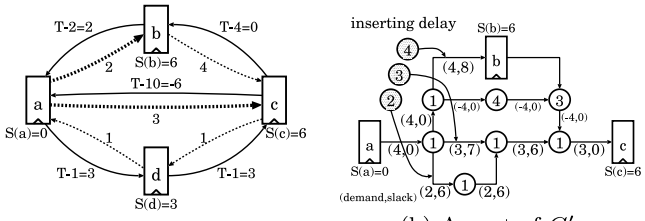


Fig. 4 Delay-slack and delay-demand.



(a) $H(G,4)$ and S'. (b) A part of G'.

Fig. 5 $H(G,4)$ of G shown in Fig. 1, clock schedule S', and circuit G' obtained by the method in [7].

the computational complexity of the problem that minimizes the amount of inserted delays in a circuit at the given clock period in g-frame is not known, we conjecture that this problem is NP-hard. We focus on the heuristic methods.

In [6], a delay insertion method that minimizes the clock period was proposed. Since the amount of an inserted delay is iteratively determined by searching the whole circuit, the method takes too much computation time. In [7], a delay insertion method that minimizes the clock period was proposed (Fig. 3). The method in [7] consists of two steps. At the first step, a clock schedule without considering all the hold constraints is determined. At the second step all the violating hold constraints of the clock schedule are recovered by delay insertion. The method in [7] determines the amount of an inserted delay according to the delay-slack and delay-demand. The delay-slack and delay-demand are defined by the difference between the arrival time of the latest signal and that of the earliest signal (see Fig. 4). If the delay-demand of an edge in a circuit is positive and a delay which is equal to the delay-demand is inserted to the edge, then the corresponding hold violation is recovered. If the delay-slack of an edge in a circuit is positive, a delay which is less than or equal to the delay-slack can be inserted to the edge without generating setup violations.

The methods in [6], [7] guarantee that the obtained circuit achieves the lower bound of the minimum clock period of the circuit G in g-frame $T_L(G)$. However, since the amount of inserted delays was not taken into account, a lot of redundant delays were inserted.

For example, the constraint graph $H(G,4)$ of G shown in Fig. 1, and the clock schedule S' which is obtained by the method in [7] are shown in Fig. 5(a). The circuit G' which is obtained by the method in [7] is shown in Fig. 5(b).

4. Proposed Method

In this paper, we propose an efficient delay insertion method that minimizes the amount of inserted delays to achieve the target clock period. Note that the circuit which achieves the target clock period is not obtained by delay insertion if the target clock period is set to be less than $T_L(G)$. Moreover, the original circuit does not need to be inserted delays if the target clock period is set to be more than or equal to $T_S(G)$. Hereafter, we assume that the target clock period is set to be more than or equal to $T_L(G)$, and less than $T_S(G)$.

The proposed method is based on the method in [7] and shown in Fig. 6. The proposed method consists of three steps. At the first step, an initial assumed clock schedule without considering all the hold constraints is obtained [7]. At the second step, the assumed clock schedule is modified to reduce the amount of inserted delays by reducing the number of hold violations. At the third step, the delays are inserted into appropriate places considering the whole circuit so that the circuit works correctly in the assumed clock schedule.

4.1 Clock Schedule Modification

In the method in [7], since an assumed clock schedule is determined ignoring hold constraints, the number of hold violations in the clock schedule tends to be large. The number of hold violations is not necessarily proportional to the amount of required delay to recover the hold violations. However, if the number of hold violations in the

Procedure **ModifySchedule**$(H(G,t), S)$
Input : constraint graph $H(G,t)$, clock schedule S
Output : clock schedule S'
Step 1 : Until no hold constraint can be recovered, repeat the following.

1. Construct $H^S(G,t)$.
2. Find an edge (a,b) on $H(G,t)$ with $\Delta(a,b) < 0$ and $dist(b,a) \geq -\Delta(a,b)$ on $H^S(G,t)$. Change $S(v) = S(v) + \min\{0, \Delta(a,b) + dist(b,v)\}$ for each vertex v.

Step 2 : return S.

Procedure **InsertDelayMinCut**(G, S)
Input : circuit G, clock schedule S
Output : circuit G' after delay insertion
Step 1 : Until all the hold constraints are satisfied, repeat the following.

1. Calculate delay-slack and delay-demand.
2. Construct the flow graph.
3. Find a minimum capacity cut.
4. Insert a delay which equals to $\min\{slack(e), demand(e)(> 0)\}$ to each edge e which belongs to the minimum cut.

Step 2 : return G.

Input : circuit G, target clock period t
Output : circuit G' after delay insertion
Step 1 : $S_{init} :=$ SchedulingZ$(H(G,t))$
Step 2 : $S_{mod} :=$ ModifySchedule$(H(G,t), S_{init})$
Step 3 : $G' :=$ InsertDelayMinCut(G, S_{mod})
Step 4 : return G'.

Fig. 6 Proposed method (S_{mod}+MinCut).

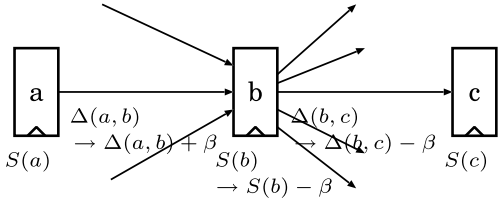
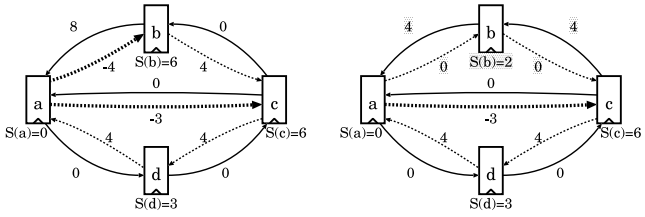


Fig. 7 The changed slack after clock schedule modification.

clock schedule is large, the amount of inserted delays usually tends to be large. The proposed method tries to minimize the number of hold violations in the assumed clock schedule to reduce the amount of inserted delays.

Let a *slack* $\Delta(a,b)$ of an edge (a,b) on $H(G,t)$ be $\Delta(a,b) = S(a) + w(a,b) - S(b)$, where $w(a,b)$ is the edge weight of (a,b) on $H(G,t)$, and S is an assumed clock schedule. Note that the timing constraint of (a,b) is violated if and only if $\Delta(a,b) < 0$. Assume that $\Delta(a,b) < 0$. The timing violation of an edge (a,b) is recovered if the clock timing of its head vertex b is decreased (see Fig. 7). However, new timing violation of an edge (b,c) for which the clock timing of its tail vertex b is decreased is generated if the slack of the edge is small. An generated new timing violation of an edge (b,c) is recovered if the clock timing of its head vertex c is decreased. By repeating the procedure in which the clock timing of head vertex is decreased, the timing violation of edge (a,b) is recovered without generating new timing violations if the clock timing of its tail vertex a is not changed by the above procedure.



(a) slacks by S'. (b) slacks by S''

Fig. 8 Slacks by clock schedule S' and S''.

Before describing the formal definition of the above procedure, we give some definitions. Let $H^S(G,t)$ be the graph obtained from $H(G,t)$ by deleting all the edges (u,v) with $\Delta(u,v) < 0$ in clock schedule S. Let the length for a path be the sum of slacks of the edges in the path in $H^S(G,t)$. Let $dist(w,x)$ be the minimum length for paths from a vertex w to x in $H^S(G,t)$. If there is no path from w to x, then let $dist(w,x)$ be infinite.

Here, we describe the procedure that reduce the number of hold violations. In the procedure, an edge (a,b) such that $\Delta(a,b) < 0$ and $dist(b,a) \geq -\Delta(a,b)$ is found and $S(v)$ is changed to $S(v) + \min\{0, \Delta(a,b) + dist(b,v)\}$ for each vertex v. By this procedure, the slack of (a,b) becomes 0 and no new violation is generated. However, the slack of an edge whose slack is negative might be decreased since the $H^S(G,t)$ does not contain these edges.

Note that the number of hold violations is decreased by the above procedure. While, the total amount of hold violations might be increased by the above procedure. In our proposed method shown in Fig. 6, a clock schedule that removes as many hold violations as possible is obtained by the above procedures and is used in delay insertion.

For example, slacks of edges on $H(G,4)$ by clock schedule S' shown in Fig. 5(a) are shown in Fig. 8(a). In Fig. 8(a), since slacks of edges (a,b) and (a,c) are negative, hold constraints (a,b) and (a,c) are violated in S'. If hold violation (a,c) is recovered by clock schedule modification, then setup constraint (c,a) is violated. Hold violation (a,c) cannot be recovered by the proposed clock schedule modification since the proposed clock schedule modification method never creates new violations. On the other hand, hold violation (a,b) can be recovered by the proposed clock schedule modification since hold violation (a,b) is recovered by decreasing $S(b)$ without generating new violations. Slacks of edges by clock schedule S'' which is obtained by the proposed clock schedule modification are shown in Fig. 8(b).

4.2 Delay Insertion

In the method in [7], since a delay is inserted iteratively without considering the global structure of the circuit, re-

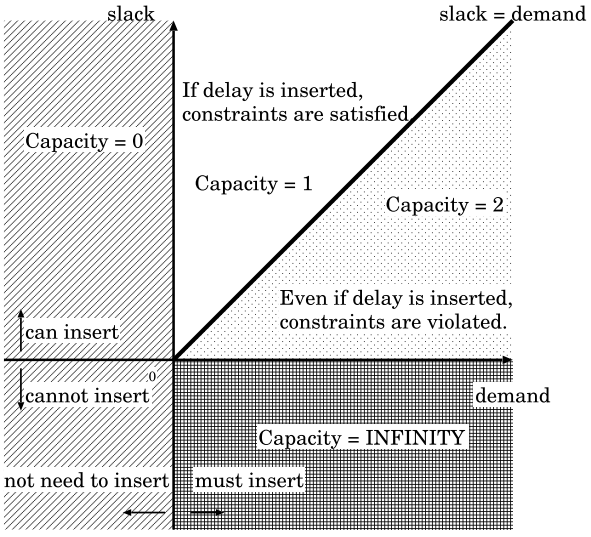


Fig. 9 Definition of the capacity.

dundant delays are inserted. If a delay is inserted into an appropriate place in the circuit, several hold violations might be recovered at once.

In the proposed method, delays are iteratively inserted into appropriate places in the circuit so that the circuit works correctly in the assumed clock schedule.

The proposed method constructs a flow graph in which the capacity of each edge, source vertices, and sink vertices, are defined. Then, delays are inserted into edges in a minimum capacity cut with finite capacity that separates source vertices and sink vertices. Source vertices, sink vertices, the capacity of each edge is defined as follows.

The source vertices are all registers which are corresponding to the tail vertices of the edges which violate the hold constraints in the constraint graph. Similarly, the sink vertices are all registers which are corresponding to the head vertices. The capacity of each edge is defined according to the delay-slack and delay-demand as shown in Fig. 9. If the delay-demand of an edge is non-positive, the capacity is set to 0. Otherwise, if the delay-slack of an edge is non-positive, the capacity is set to infinite. Otherwise, if the delay-slack of an edge is larger than or equal to the delay-demand, the capacity is set to one. Otherwise, if the delay-slack of an edge is smaller than the delay-demand, the capacity is set to two.

If the delay-demand of an edge is non-positive, the delay does not need to be inserted to the edge. The capacities of these edges are set to 0 so that the minimum capacity cut does not take these edges into account. If the delay-slack of an edge is non-positive, the delay cannot be inserted to the edge. The capacities of these edges are set to infinite so that the minimum capacity cut does not contain these edges. If the delay-slack of an edge is larger than or equal to the delay-demand, a hold violation is recovered by inserting delay to the edge only. The capacities of these edges are set to one so that the minimum capacity cut contains these edges easier. If the delay-slack of an edge is smaller than

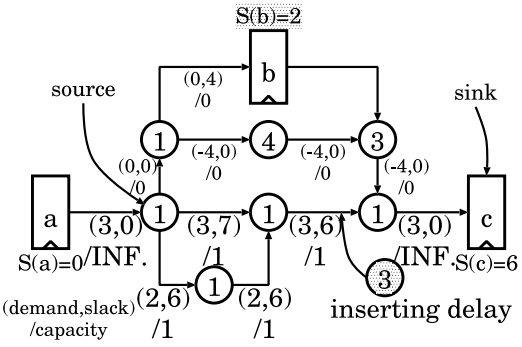


Fig. 10 The flow graph for G shown in Fig. 1 whose clock schedule is set to S'' shown in Fig. 8.

the delay-demand, then a hold violation is not recovered by inserting delay to the edge only although the delay can be inserted to the edge. The capacities of these edges are set to two.

By delay insertion into edges, no setup violation is generated if the number of edges to which delays are inserted is at most one for any register-to-register path in G and the amount of inserted delay to an edge is at most its delay-slack. Basically, the delays are inserted into the edges on the found minimum cut of the flow graph defined above. However, there are two exceptions.

The first one is the amount of delay insertion. In some cases, the found minimum cut contains more than one edge of a register-to-register path. In such cases, no delay is inserted into an edge if some delays are inserted into edges on the source side of the edge.

The second one is the construction of the flow graph. In some cases, there is a path from a source a to a sink b in which the delay-slack of every edge is 0 and the capacity is infinite in the flow graph defined above. In such cases, we do not need to insert delays into edges on the path since it is guaranteed that the hold constraint of (a, b) is not violated [7], but a and b are defined as source and sink due to other source-to-sink paths, respectively. In order to ignoring the path from a to b, the flow graph is modified as follows. An edge with infinite capacity whose tail vertex is a source is removed. Furthermore, for each removed edge, the tail vertex is removed from source vertices if it has no outgoing edge and the head vertex is added into source vertices if it has no incoming edge.

If every source to sink path has exactly one edge with capacity one in the found minimum cut, then all the hold violations are recovered at once. Otherwise, the procedure is repeated.

For example, the flow graph for G shown in Fig. 1 whose clock schedule is set to S'' shown in Fig. 8(b) is shown in Fig. 10. The delay shown in Fig. 10 is inserted by the proposed method. The amount of inserted delays by the proposed method is less than that by the method [7] shown in Fig. 5.

5. Experimental Results

We implement the clock scheduling method of the method in [7] (S_{init}), the delay insertion method of the method in [7] (DFS), the clock scheduling method of the proposed method (S_{mod}), and the delay insertion method of the proposed method (MinCut), respectively. We implement these methods in C++, which was compiled by gcc4.1.2, and execute on a PC with a 2.93 GHz Intel Core2 CPU and 2 GB RAM. In order to observe the efficiency of each clock scheduling method and that of each delay insertion method, we apply four methods to circuits in combination: S_{init}+DFS [7], S_{init}+MinCut, S_{mod}+DFS, and S_{mod}+MinCut (proposed method). We also implement the mixed integer linear programming formulation for minimization of the inserted delay (MILP) based on [8], [9]. MILP is solved by CPLEX11.0.0 [10] on the same PC. We perform these methods on the ISCAS89 benchmark suite.

First, the delay of each NOT, AND, OR, NAND, and NOR gate is set to 1 and that of each register and routing is set to 0. In experiments, we assume that delay insertion is realized by inserting delay gates and we evaluate the delay insertion methods according to the number of inserted delay gates and the computation time. The delay of each inserted delay gate is also set to 1. In 30 circuits among 48 ISCAS89 benchmark circuits, since the minimum clock periods in g-frame are not decreased by the delay insertion, the original circuits are optimal. The other 18 circuits are shown in Table 1. In this experiment, the target clock period is set to the lower bound of minimum clock period in g-frame T_L [6], [7].

Table 1 Benchmark circuits and those clock periods under the delay of each gate is set to 1.

	size				clock period		
circuit	$\|V_g\|$	$\|E_g\|$	$\|V_r\|$	$\|E_r\|$	T_C	T_S	T_L
s298	270	400	15	168	9	6	5.334
s344	360	479	16	230	20	17	14.000
s349	362	484	16	230	20	17	14.000
s444	408	584	22	346	11	7	6.584
s526	432	689	22	330	9	6	5.500
s635	639	829	33	1184	127	124	66.000
s991	1142	1457	20	218	59	55	54.000
s1269	1231	1718	38	702	35	30	18.667
s1423	1480	1991	75	3794	59	54	53.000
s1512	1704	2208	58	1226	30	24	22.500
s3271	3403	4541	117	1880	28	19	14.715
s3384	3780	4875	184	3876	60	51	26.500
s4863	4942	6707	105	1530	58	53	30.000
s6669	6722	9039	240	4816	93	81	25.167
prolog	3511	4786	137	1530	26	14	12.500
s15850	20753	24712	598	31782	82	57	42.000
s15850.1	20690	24712	535	24926	82	71	63.000
s35932	35622	48145	1729	13880	29	28	27.000

$|V_g|$ the number of vertices in the circuit graph
$|E_g|$ the number of directed edges in the circuit graph
$|V_r|$ the number of register in the circuit
$|E_r|$ the number of directed edges in the constraint graph
T_C the minimum clock period of the original in c-frame
T_S the minimum clock period of the original in g-frame
T_L the lower bound clock period in g-frame

The experimental results of MILP and four combinational methods are shown in Table 2. The optimum solutions except s6669 are obtained by MILP. On the other hand, the optimum solution of s6669 is not obtained by MILP due to out of memory when the MILP is solved by CPLEX. The comparisons between S_{init}+DFS and S_{mod}+DFS, and between S_{init}+MinCut and S_{mod}+MinCut show that the proposed clock schedule modification method reduces the number of inserted delay gates and the computation time. Furthermore, the comparisons between S_{init}+DFS and S_{init}+MinCut, and between S_{mod}+DFS and S_{mod}+MinCut show that the proposed delay insertion method also reduces the number of inserted delay gates and the computation time. The number of inserted delay gates obtained by the proposed method S_{mod}+MinCut is only about 1.24 times as large as that of the optimum solution on average, and the computation time of the method in [7] is about four times as fast as that of MILP on average. In particular, the optimum solutions are obtained by the proposed method in 12 circuits among 18 circuits.

Next, the delay of each NOT gate is set to 1, that of each NAND, NOR and inserted delay gate is set to 2, that of each AND and OR gate is set to 3, and that of each register and routing is set to 0. In 23 circuits among 48 ISCAS89 benchmark circuits, the original circuits are optimal. The other 25 circuits and the result of MILP, S_{init}+DFS [7], and S_{mod}+MinCut (proposed method) are shown in Table 3. In this experiment, the target clock period is also set to the lower bound of minimum clock period in g-frame T_L.

The optimum solutions of three circuits are not obtained by MILP under this delay model. We expect that the optimum solutions are not obtained by MILP for larger circuit under more practical delay model. While, the proposed method obtain optimum solutions in short time in most circuits.

Last, in order to observe the relations between the target clock period and the number of inserted delay gates, the target clock period is changed from T_L to T_S of the original circuit. Note that the optimum solution of inserted delays is 0 when the target clock period is set to T_S. The delay of each NOT gate is set to 1, that of each NAND, NOR and inserted delay gate is set to 2, that of each AND and OR gate is set to 3, and that of each register and routing is set to 0. In the experiments, MILP and four combinational methods are applied to s3271 and s3384. The results of s3271 and s3384 are shown in Fig. 11 and Fig. 12, respectively.

In two circuits, if the assumed clock schedule is not modified, then the delays are inserted when the target clock period is set to T_S. This fact means that the assumed clock schedule obtained by the method in [7] is not appropriate. On the other hand, if the assumed clock schedule is modified by the proposed method, the number of inserted delay gates is 0 when the target clock period is set to T_S. The number of inserted delay gates obtained by MILP is not increased when the target clock period is increasing. While, in Fig. 12, the number of inserted delay gates obtained by

Table 2 Experimental results under the delay of each gate is set to 1.

	MILP		S_{init}+DFS [7]		S_{init}+MinCut		S_{mod}+DFS		S_{mod}+MinCut	
circuit	$\#D_{ins}$	Time[s]	$\#D_{ins}$	Time[s]	$\#D_{ins}$	Time[s]	$\#D_{ins}$	Time[s]	$\#D_{ins}$	Time[s]
s298	3	0.01	13	<0.01	8	<0.01	3	<0.01	3	<0.01
s344	3	0.01	69	<0.01	69	<0.01	3	<0.01	3	<0.01
s349	3	0.01	69	<0.01	69	<0.01	3	<0.01	3	<0.01
s444	13	0.02	20	<0.01	17	<0.01	14	<0.01	13	<0.01
s526	3	0.03	13	<0.01	8	<0.01	3	<0.01	3	<0.01
s635	422	0.03	1304	0.01	1304	<0.01	422	<0.01	422	<0.01
s991	1	0.05	6293	0.13	1649	0.02	1	0.01	1	0.01
s1269	120	0.85	615	0.04	511	0.02	313	0.03	197	0.01
s1423	1	0.05	3090	0.16	2825	0.03	1	0.02	1	0.02
s1512	4	0.07	1395	0.17	1024	0.04	10	0.01	4	0.01
s3271	48	0.41	1668	0.53	844	0.04	68	0.09	54	0.02
s3384	256	1.68	2309	0.33	2019	0.03	433	0.14	386	0.02
s4863	552	0.41	6229	2.88	4535	0.81	2145	1.16	1257	0.31
s6669	*2488	*490.59	11465	6.62	7238	1.18	8492	4.87	5615	1.02
prolog	9	0.22	514	0.35	138	0.02	16	0.03	9	0.01
s15850	82	1.48	10495	30.11	4926	1.25	320	3.03	120	0.84
s15850.1	8	0.96	21785	35.55	14150	2.32	8	0.36	8	0.35
s35932	1	2.03	4290	95.03	4290	3.41	1	0.42	1	0.42
AVE. [%]	(100.00)	(100.00)	94744.76	945.50	60893.00	72.67	173.74	55.71	123.73	22.98

$\#D_{ins}$ the number of inserted delay gates to achieve T_L
Time[s] computation time
AVE. average of comparison ratio between heuristic and optimal except computation time which is less than 0.01[s]
∗ The best feasible solution of s6669 is shown before out of memory.

Table 3 Experimental results under the delay of each NOT gate is set to 1, the delay of each NAND, NOR, and inserted gate is set to 2, and the delay of each AND and OR gate is set to 3.

	clock period			MILP		S_{init}+DFS [7]		S_{mod}+MinCut	
circuit	T_C	T_S	T_L	$\#D_{ins}$	Time[s]	$\#D_{ins}$	Time[s]	$\#D_{ins}$	Time[s]
s298	18	12	10.000	1	0.01	14	<0.01	1	<0.01
s344	38	34	29.000	3	0.01	81	<0.01	3	<0.01
s349	38	34	29.000	3	0.01	81	<0.01	3	<0.01
s382	18	12	11.250	1	0.02	9	<0.01	1	<0.01
s400	18	12	11.250	1	0.01	9	<0.01	1	<0.01
s444	20	13	11.667	8	0.07	12	<0.01	8	<0.01
s526	18	12	11.000	1	0.04	9	<0.01	1	<0.01
s526n	18	12	11.000	1	0.04	9	<0.01	1	<0.01
s635	162	158	88.500	233	0.18	774	0.01	233	0.01
s991	117	110	109.000	1	0.04	6350	0.12	1	0.01
s1269	70	61	39.334	*126	*990.29	653	0.04	197	0.01
s1423	164	156	146.000	5	0.07	4498	0.16	5	0.02
s1512	54	43	40.500	3	0.10	944	0.10	3	<0.01
s3271	58	34	27.715	41	1.82	1774	0.54	43	0.02
s3330	66	40	32.000	9	0.52	233	0.17	10	<0.01
s3384	168	147	75.500	406	227.67	3762	0.36	611	0.03
s4863	144	129	75.000	*623	*396.68	7188	2.36	1495	0.21
s6669	231	197	62.167	*3054	*623.91	13767	5.48	6466	0.74
s9234	107	72	63.000	13	0.73	3948	4.33	18	0.04
s9234.1	107	72	63.000	13	0.98	4136	5.35	18	0.04
prolog	68	40	31.000	11	0.43	543	0.30	11	0.01
s13207	106	76	75.000	1	0.83	16442	12.50	1	0.11
s15850	141	104	78.000	49	3.40	10349	31.74	71	0.56
s15850.1	141	124	103.000	14	1.35	17403	36.11	14	0.47
s38417	85	61	60.000	1	2.34	52562	340.43	1	1.24
AVE.[%]				(100.00)	(100.00)	315608.90	1268.80	119.85	12.66

$\#D_{ins}$ the number of inserted delay gates to achieve T_L
Time[s] computation time
AVE. average of comparison ratio between heuristic and optimal except computation time which is less than 0.01[s]
∗ The best feasible solutions of s1269, s4863, and s6669 are shown before out of memory.

the proposed method when the target clock period is set to 130 are larger than that when the target clock period is set to 125. The clock schedule obtained by the proposed clock scheduling modification method depends on the given target clock period, and the number of inserted delay gates obtained by the proposed delay insertion method depends on

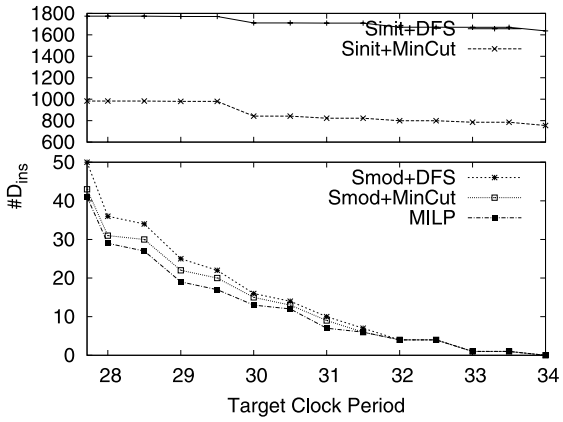


Fig. 11 The result of s3271.

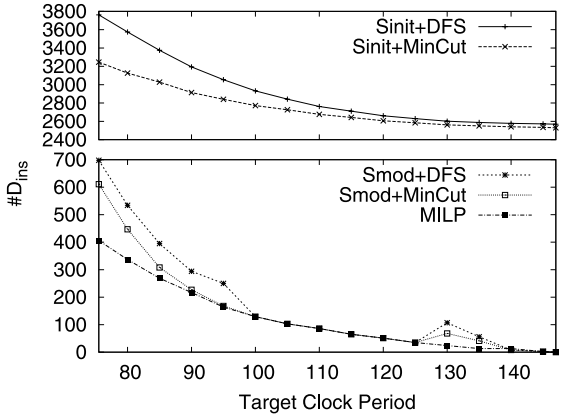


Fig. 12 The result of s3384.

the assumed clock schedule. Since both the proposed clock scheduling modification method and the proposed delay insertion method are heuristic, the number of inserted delay gates obtained by the proposed method is rarely increased when the target clock period is increasing. However, the number of inserted delay gates obtained by the proposed method is optimal in many target clock periods.

It is known that MILP can be solved in short computation time if the constraint matrix of a MILP formulation is totally unimodular. The constraint matrix of MILP formulation for minimization of the inserted delay gates is not totally unimodular, but seems to have a similar property when the target clock period is integer and the input variables are small integers. Then, the computation time of MILP for the delay insertion is very fast in above experiments. However, it seems that MILP based delay insertion methods are not applicable to the problems which are defined based on a more practical delay model. For example, the delay of each NOT, NAND, NOR, AND, OR, inserted delay gates, register, and wire are set to 1001, 1999, 2001, 2999, 3001, 2001, 0, and 0, respectively. Then, CPLEX stops due to out of memory and gives no feasible solutions in some circuits. On the other hand, our proposed method gives a feasible solution in a short time in all circuits. Our method is expected to be applicable to larger circuits even if a practical delay model is adopted.

6. Conclusion

We propose an efficient delay insertion method that minimizes the amount of inserted delays based on the method in [7]. Experiments showed that the proposed method can obtain optimum solutions in short time in many cases under the assumption that the maximum delay of each element is equal to its minimum delay.

For the future work, we will proposed more efficient delay insertion method according to the amount of inserted delays and the computation time, apply the proposed method to the real delay model and evaluate the whole circuit performance after the delay insertion including the clock distribution circuit.

Acknowledgments

This research was partially supported by Japan Society for the Promotion of Science (JSPS), Grant-in-Aid for JSPS Fellows 19·6015, 2007.

References

[1] Y. Kohira, S. Tani, and A. Takahashi, “Clock scheduling method and delay insertion method for minimization of inserted delay,” 21st Workshop on Circuits and Systems in Karuizawa, pp.629–634, 2008.
[2] J. Fishburn, “Clock skew optimization,” IEEE Trans. Comput., vol.39, no.7, pp.945–951, 1990.
[3] R.B. Deoker and S.S. Sapatneker, “A graph-theoretic approach to clock skew optimization,” ISCAS, pp.407–410, 1994.
[4] A. Takahashi and Y. Kajitani, “Performance and reliability driven clock scheduling of sequential logic circuits,” ASP-DAC’97, pp.37–43, 1997.
[5] A. Takahashi, “General synchronous circuits using global clock — Design methodologies, tools, and prospects,” IPSJ SIG Technical Report, 2006-SLDM-126, vol.2006, no.111, pp.159–164.
[6] T. Yoda and A. Takahashi, “Clock period minimization of semi-synchronous circuits by gate-level delay insertion,” IEICE Trans. Fundamentals, vol.E82-A, no.11, pp.2383–2389, Nov. 1999.
[7] Y. Kohira and A. Takahashi, “Clock period minimization method of semi-synchronous circuits by delay insertion,” IEICE Trans. Fundamentals, vol.E88-A, no.4, pp.892–898, April 2005.
[8] B. Taskin and I.S. Kourtev, “Delay insertion method in clock scheduling,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.25, no.4, pp.651–663, 2006.
[9] C. Lin and H. Zhou, “Clock skew scheduling with delay padding for prescribed skew domains,” ASP-DAC, pp.541–546, 2007.
[10] ILOG, CPLEX, http://www.ilog.com/

Yukihide Kohira received his B.E., M.E., and D.E. degrees from Tokyo Institute of Technology, Tokyo, Japan, in 2003, 2005, and 2007, respectively. He is currently a researcher of Department of Communications and Integrated Systems in Tokyo Institute of Technology. His research interests are in VLSI design automation and combinational algorithms. He is a member of IEEE and IPSJ.

Shuhei Tani received his B.E. degree from Tokyo Institute of Technology, Tokyo, Japan, in 2007. He is currently pursuing the M.E. degreer of Department of Communications and Integrated Systems in Tokyo Institute of Technology. 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 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 visited University of California, Los Angeles, U.S.A., as a visiting scholar from 2001 to 2002. He is currently with Department of Communications and Integrated Systems, Graduate School of Science and Engineering, Tokyo Institute of Technology. His research interests are in VLSI layout design and combinational algorithms. He is a member of IEEE and IPSJ.