論文 / 著書情報
Article / Book Information

Title	Clock Period Minimization Method of Semi-Synchronous Circuits by Delay Insertion
Authors	Yukihide Kohira, Atsushi Takahashi
Citation	IEICE Trans. Fundamentals, Vol. E88-A, No. 4, pp. 892-898
Pub. date	2005, 4
URL	http://search.ieice.org/
Copyright	(c) 2005 Institute of Electronics, Information and Communication Engineers

IEICE TRANS. FUNDAMENTALS, VOL.E88–A, NO.4 APRIL 2005

PAPER Special Section on Selected Papers from the 17th Workshop on Circuits and Systems in Karuizawa

Clock Period Minimization Method of Semi-Synchronous Circuits by Delay Insertion

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

SUMMARY Under the assumption that clock can be inputted to each register at an arbitrary timing, the minimum feasible clock period can be determined if delays between registers are given. This minimum feasible clock period might be reduced if delays between some registers are increased by delay insertion. In this paper, we propose a delay insertion algorithm to reduce the minimum clock period. First, the proposed algorithm determines a clock schedule ignoring some constraints. Second, the algorithm inserts delays to recover ignored constraints according to the delay-slack and delay-demand of the obtained clock schedule. We show that the proposed algorithm achieves the minimum clock period by delay insertion if the delay of each element in the circuit is unique. Experiments show that the amount of inserting delay and computational time are smaller than the conventional algorithm.

key words: *delay insertion, clock period minimization, semi-synchronous circuit, delay-slack, delay-demand*

1. Introduction

The semiconductor manufacturing process technology has improved the scale, speed and power consumption of LSI circuits. However, increasing ratio of routing delay in signal propagation delay makes the simultaneous clock distribution of every register difficult. The increases of size and power consumption of a clock distribution circuit have become serious issues in conventional synchronous circuits. While, a semi-synchronous circuit, in which the clock is assumed to be distributed periodically to each individual register though not necessarily to all registers simultaneously, is expected to give an essential solution. By using semi-synchronous framework, the improvements of clock frequency, clock distribution circuit size, peak power consumption, and etc. are expected to be achieved.

In recent studies of semi-synchronous circuit, clock scheduling algorithms [1] and clock distribution circuit synthesis algorithms [2] for given logic circuits were proposed. However given logic circuits are synthesized for complete-synchronous framework. In order to improve the clock period in complete-synchronous framework, the circuits are synthesized so that the maximum delay between registers is as small as possible. Thus, the size of a module tends to be large, since it is synthesized under the tighter delay constraints. Larger transistors and wider wires are used to reduce the delay. Moreover, a number of buffers is required in recent technology to reduce the delay of a long interconnect. However, in semi-synchronous framework, the clock period might not be reduced even if the maximum delay is reduced. The effort in complete-synchronous framework might degrade the circuit performance in semi-synchronous framework. So the optimization of circuit synthesis that takes semi-synchronous framework into account must be investigated.

As logic circuit improvement methods for semi-synchronous circuits, the delay insertion method was proposed in [3] and the gate sizing method was proposed in [4]. The algorithm in [3] achieves the minimum clock period in semi-synchronous framework which can be achieved by delay insertion, but it cannot be applied to large circuits since it inserts too much delay and takes too much time. While, the algorithm in [4] does not necessarily achieve the minimum clock period by delay insertion.

In this paper, we propose a delay insertion algorithm to reduce the minimum clock period of a circuit in semi-synchronous framework, especially for the circuit synthesized in complete-synchronous framework. The proposed algorithm will remove the resources added in synthesis in complete-synchronous framework which degrade the performance in semi-synchronous framework. The proposed algorithm increases delays as small as possible to reduce the minimum clock period in semi-synchronous framework. Since the amount of inserting delays by the proposed algorithm is small, the improved circuits are obtained by small modifications. 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. The area of the obtained circuit will be almost the same as the original circuit.

The proposed algorithm consists of two stages. First, the proposed algorithm determines a clock schedule that achieves the minimum clock period by ignoring some constraints. Second, the algorithm inserts delays to recover ignored constraints according to the delay-slack and delay-demand of the obtained clock schedule. We show that the proposed algorithm achieves the minimum clock period by delay insertion if the delay of each element in the circuit is unique (that is, maximum delay = minimum delay for each element). Experiments show that the amount of inserting

Manuscript received June 29, 2004.
Manuscript revised October 6, 2004.
Final manuscript received December 2, 2004.

[†]The authors are with the Department of Communications and Integrated Systems, Tokyo Institute of Technology, Tokyo, 152-8552 Japan.
a) E-mail: kohira@lab.ss.titech.ac.jp

DOI: 10.1093/ietfec/e88–a.4.892

delay and computational time are smaller than the conventional algorithm [3].

2. Preliminaries

In this paper, we consider a circuit consisting of registers and gates, and wires connecting them. We refer to them as elements. A circuit is represented by the *circuit 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 each element has a unique delay. Let $d(v)$ be the weight of $v \in V_g$ which corresponds to the delay of corresponding element. Let $d(e) = d(u)$, where $e = (u, v) \in E_g$ $(u, v \in V_g)$. This means that an edge weight is equal to the weight of the head vertex of the edge. Let $d(P) = \sum_{k=1}^{i} d(e_k)$, where $P = (e_1, e_2, \ldots, e_i)$ is a path in G.

Let V_r be the register set of G. Necessarily, the register set is a subset of V_g. An example of the circuit graph is shown in Fig. 1. In Fig. 1, $\{a, b, c, d\}$ is the register set, and the figure in each vertex represents its weight.

In semi-synchronous circuits, the clock input 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.

A circuit works correctly with clock period T if the following two types of constraints are satisfied for every register pair with signal propagations [5].

Setup (No-Zero-Clocking) Constraints

$$S(r_i) - S(r_j) \leq T - d_{max}(r_i, r_j)$$

Hold (No-Double-Clocking) Constraints

$$S(r_j) - S(r_i) \leq d_{min}(r_i, r_j)$$

where $d_{max}(r_i, r_j)$ is the maximum delay and $d_{min}(r_i, r_j)$ is the minimum delay from register r_i to r_j (Fig. 2).

Since complete-synchronous circuits have the premise that a clock ticks all the register simultaneously, the clock period must be larger than the maximum delay between registers. On the other hand, semi-synchronous 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.

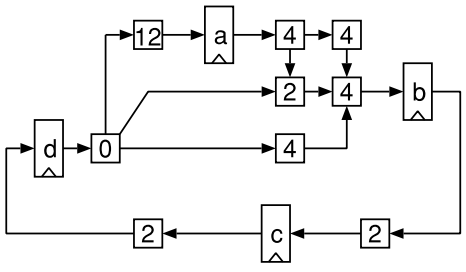


Fig. 1 Circuit graph G.

Let $T_S(G)$ be the minimum clock period of a circuit G in semi-synchronous framework under the assumption that clock can be inputted to each register at an arbitrary timing. Hereafter, we simply call $T_S(G)$ the *minimum clock period* of G. $T_S(G)$ is determined by the *constraint graph* $H(G, T)$ for G with clock period T defined as follows. The vertex set V_r of $H(G, T)$ corresponds to registers in G. The directed edge set A_r of $H(G, T)$ corresponds to two types of constraints. An edge from r_i to r_j with $d_{\min}(r_i, r_j)$, called the D-edge, corresponds to the Hold constraint, and an edge from r_j to r_i with $T - d_{\max}(r_i, r_j)$, called the Z-edge, corre-

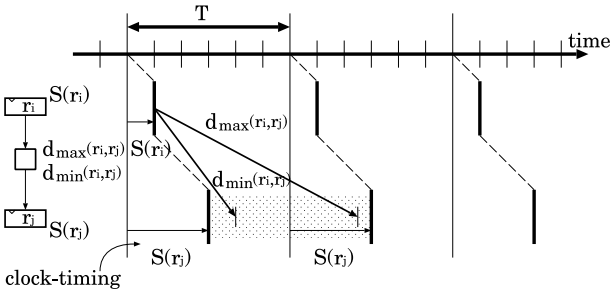
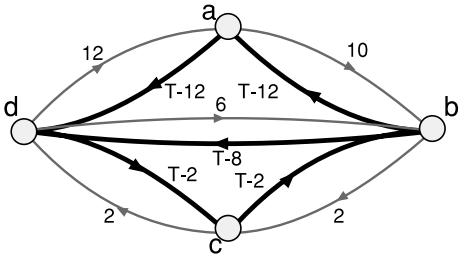
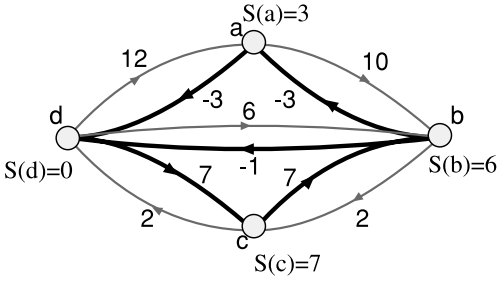


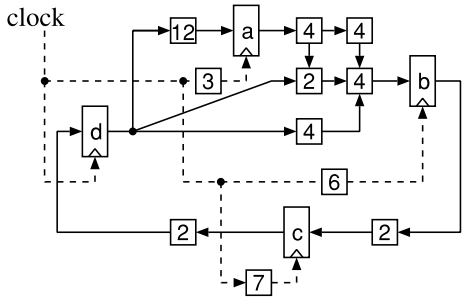
Fig. 2 Timing chart.



(a) Constraint graph $H(G, T)$.

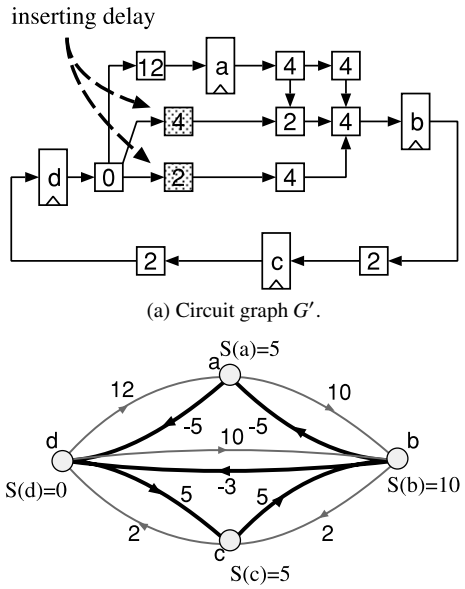


(b) Constraint graph $H(G, 9)$.



(c) Semi-synchronous circuit with clock circuit.

Fig. 3 Semi-synchronous circuit and its constraint graphs.



(a) Circuit graph G'.

(b) Constraint graph $H(G', 7)$.

Fig. 4 Circuit after delay insertion to G in Fig. 1.

sponds to the Setup constraint.

Let the weight of a directed cycle in the constraint graph be the sum of edge weights on the directed cycle. We refer to the cycle whose weight is 0 and negative as *zero-cycle* and *negative-cycle*, respectively.

Theorem 1 ([5], [6]): $T_S(G)$ is the minimum t such that there is no negative-cycle in the constraint graph $H(G, t)$.

For example, the delay from register d to a in G in Fig. 1 is 12 which is the maximum delay between registers. So the minimum clock period in complete-synchronous framework is 12. The constraint graph $H(G, T)$ for G in Fig. 1 is shown in Fig. 3(a). Since $H(G, 9)$ shown in Fig. 3(b) includes no negative-cycle and the cycle (a, d, b) is a zero-cycle, the minimum clock period $T_S(G)$ of G is 9. The semi-synchronous circuit that achieves the clock period 9 is shown in Fig. 3(c).

Delay insertion to a circuit is represented by replacement of an edge with a series of two edges with a positive weight vertex. The circuit G' obtained from G in Fig. 1 by inserting delays is shown in Fig. 4(a). In the constraint graph $H(G', T)$, only edge weights are changed from $H(G, T)$ according to the delay insertion. Since $H(G', 7)$ shown in Fig. 4(b) includes no negative-cycle, and the cycles (a, d, c, b) and (a, d, b) in $H(G', 7)$ are zero-cycles, the minimum clock period $T_S(G')$ is 7. In this case, the minimum clock period is improved by delay insertion.

3. Previous Algorithm

3.1 A Lower Bound of the Clock Period

If a circuit does not work correctly with a clock period t, all the negative-cycles in the constraint graph should be eliminated so that the circuit works correctly. Edge weights should be increased in order to eliminate all the negative-cycles. Since D-edge weight is d_{min} and Z-edge weight is $T - d_{max}$, we should increase minimum delays or reduce maximum delays so that edge weights are increased.

In semi-synchronous circuits, the lower bound of the clock period that can be achieved by delay insertion is equal to the *maximum delay-to-register ratio* $T_B(G)$ [3].

Definition 1 ([3]): The maximum delay-to-register ratio is defined as

$$T_B(G) = \max_{L \in \text{all cycles in } G} \frac{D(L)}{N(L)},$$

where $N(L)$ is the number of registers in directed cycle L in G and $D(L)$ is the weight of L.

We refer to the maximum delay-to-register ratio $T_B(G)$ as the *minimum clock period by delay insertion.*

In complete-synchronous framework, $T_B(G)$ can be achieved if an arbitrary amount of retiming is allowed. However, the number of registers might be increased, and gate decompositions might be required.

Theorem 2 ([3]): If $T_S(G) > T_B(G)$, every zero-cycle in $H(G, T_S(G))$ contains at least one D-edge.

Therefore, the minimum clock period is improved by delay insertion to the circuit if $T_S(G) > T_B(G)$.

3.2 Delay-Slack

The maximum delay-to-register ratio $T_B(G')$ of the circuit G' which is obtained from a circuit G by inserting delays must not become greater than $T_B(G)$ so that G' achieves the lower bound of the clock period of G. $T_B(G')$ does not change from $T_B(G)$ if $D(L)$ of any cycle L in G' is at most $T_B(G) \cdot N(L)$. For each edge of G, the *delay-slack* [3] is defined as the maximum delay insertion that keeps the minimum clock period by delay insertion.

Definition 2 (delay-slack): For a directed cycle L in G, the *cycle-slack* is defined as

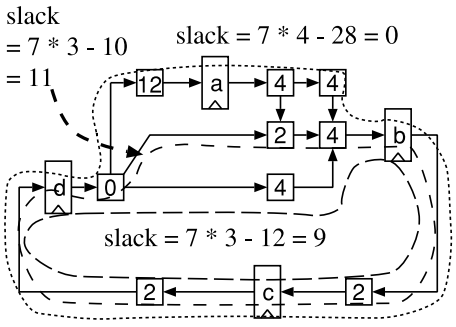
$$\text{cycle-slack} = T_B(G) \cdot N(L) - D(L).$$

The delay-slack of an edge (u, v) in G is the minimum cycle-slack over all cycles that contain (u, v).

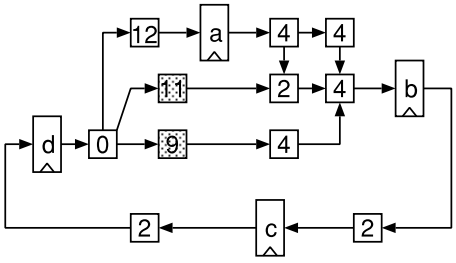
The delay insertion to (u, v) less than or equal to the delay-slack of (u, v) keeps the minimum clock period by delay insertion.

In the algorithm in [3], the delay equal to delay-slack of an edge is inserted to the edge when the edge belongs to the minimum delay path from r_i to r_j where (r_i, r_j) is a D-edge in a negative-cycle in $H(G, T_B(G))$. The delay-slack of each edge is recalculated after each delay insertion. For example, cycle-slacks of G in Fig. 1 are determined as shown in Fig. 5(a). The circuit obtained from G by the algorithm in [3] is shown in Fig. 5(b).

The computation time of this algorithm is $O(ngp^2)$, where n is the number of vertices in the constraint graph, and g and p are the numbers of vertices and directed edges in the circuit graph, respectively.



(a) Cycle-slacks of G ($T_B(G) = 7$).



(b) Circuit G''.

Fig. 5 Circuit obtained from G in Fig. 1 by the algorithm in [3].

4. Proposed Algorithm

In the proposed delay insertion algorithm, the clock timing of each register is determined by Setup constraints before delay insertion, and delays are inserted into subcircuits to satisfy Hold constraints. We show that the circuit which works correctly when the clock period is the minimum clock period by delay insertion is obtained in short time.

4.1 Scheduling

We determine the clock timing of each register from the constraint graph consisting of only Z-edges with the clock period $T = T_B(G)$ by clock-scheduling algorithm [6].

4.2 Delay Insertion

The clock schedule from Setup constraints may not satisfy Hold constraints. So delays are inserted in order to satisfy Hold constraints while keeping Setup constraints.

For any edge $e \in E_g$, let $break(e)$ be the time when the earliest signal arrives at e, $complete(e)$ be the time when the latest signal arrives at e, $hold(e)$ be the earliest possible time of signal arrival at e that can keep Hold constraints, and $setup(e)$ be the latest possible time of signal arrival at e that can keep Setup constraints. Moreover, let *delay-slack* $slack(e)$ be $setup(e) - complete(e)$, that means the amount of delay which can be inserted, and *delay-demand* $demand(e)$ be $hold(e) - break(e)$, that means the amount of delay which must be inserted (Fig. 6). The above definitions are stated formally as follows.

Definition 3: For a given clock schedule S of a circuit G,

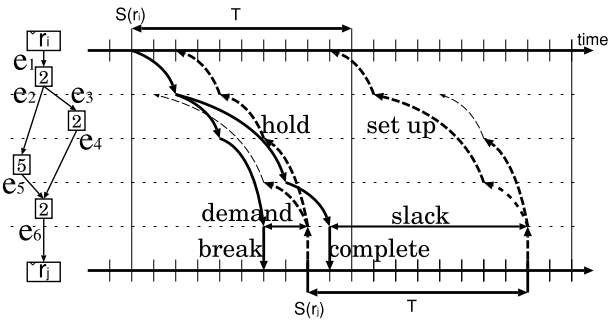


Fig. 6 *break*, *complete*, *hold*, *setup*, *slack*, *demand*.

break, *complete*, *hold*, *setup*, *slack*, and *demand* of $e = (u, v) \in E_g$ are defined as follows; If u is a register ($u \in V_r$),

$$break(e) = S(u) + d(e),$$
$$complete(e) = S(u) + d(e),$$

otherwise,

$$break(e) = \min_{e' \in Fi(e)} \{break(e') + d(e)\},$$
$$complete(e) = \max_{e' \in Fi(e)} \{complete(e') + d(e)\},$$

where $Fi(e)$ is the edge set whose tail is u. If v is a register ($v \in V_r$),

$$hold(e) = S(v),$$
$$setup(e) = S(v) + T,$$

otherwise,

$$hold(e) = \max_{e' \in Fo(e)} \{hold(e') - d(e')\},$$
$$setup(e) = \min_{e' \in Fo(e)} \{setup(e') - d(e')\},$$

where $Fo(e)$ is the edge set whose head is v. Moreover,

$$slack(e) = setup(e) - complete(e),$$
$$demand(e) = hold(e) - break(e).$$

Let $e' = (t, u)$ and $e = (u, v)$ be the edges in E_g such that $u \notin V_r$. If $break(e) = break(e') + d(e)$, we say that e' determines *break* of e or *break* of e depends on e'. Let $P = (e_1, e_2, \ldots, e_{n-1})$ be a path in G such that $e_i = (v_{i-1}, v_i)$, $v_0 \in V_r$, and $v_i \notin V_r (1 \le i \le n-1)$. If e_i depends on $e_{i-1} (1 \le i \le n)$, then we say that P determines *break* of e_n or *break* of e_n depends on P. If P determines *break* of e_n, $break(e_n) = S(v_0) + \sum_{k=1}^{n} d(e_k) = S(v_0) + d(P) + d(e_n)$. For example, *break* of e_6 does not depend on e_5 but depends on e_4 in Fig. 6. The path (e_1, e_3, e_4) determines *break* of e_6. Similarly, dependency is defined for *complete*, *hold*, and *setup*.

Theorem 3: A clock schedule S with clock period T satisfies Setup constraints if and only if $slack(e) \ge 0$ for all edges e in E_g.

Proof. Let e be an edge whose *complete* depends on a path

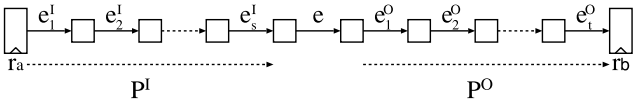


Fig. 7 The paths that determine *complete* and *setup* of e.

$P^I = (e^I_1, e^I_2, \ldots, e^I_s)$ and whose *setup* depends on a path $P^O = (e^O_1, e^O_2, \ldots, e^O_t)$, where the head of e^I_1 is the register r_a and the tail of e^O_t is the register r_b (Fig. 7). Note that

$$complete(e) = S(r_a) + d(P^I) + d(e),$$
$$setup(e) = S(r_b) + T - d(P^O),$$
$$d_{max}(r_a, r_b) \geq d(P^I) + d(e) + d(P^O).$$

So we have

$$\begin{aligned} slack(e) &= setup(e) - complete(e) \\ &= S(r_b) - S(r_a) + T - (d(P^I) + d(e) + d(P^O)) \\ &\geq S(r_b) - S(r_a) + T - d_{max}(r_a, r_b). \end{aligned}$$

If $setup(e) < 0$, we have

$$S(r_a) - S(r_b) > T - d_{max}(r_a, r_b).$$

This contradicts the assumption that the clock schedule S satisfies Setup constraints.

The sufficiency can be proved similarly. □

Even if delay less than or equal to the amount of $slack(e)$ is inserted to e, Setup constraints are satisfied.

Following theorem can be proved similarly.

Theorem 4: A clock schedule S with clock period T satisfies Hold constraints if and only if $demand(e) \leq 0$ for all edges e in E_g.

So we must insert delay so that $demand(e) \leq 0$ for all edges e in order to satisfy Hold constraints.

Assume that Hold constraint of e is violated. If the delay equal to $\min\{slack(e), demand(e)\}$ is inserted to e, then Setup constraints are not violated while Hold constraints are relaxed.

4.3 Algorithm

We propose a delay insertion algorithm as follows.

Inputs : circuit graph G
Outputs : circuit graph G' after delay insertion

Step 1 : Determining the clock timing of each register from the constraint graph consisting of Z-edges with the clock period $T = T_B(G)$.

Step 2 : Until an edge e whose $demand(e) > 0$ exists, repeat the following. Insert a delay equal to $\min\{slack(e), demand(e)\}$ to e. Recalculate delay-slack and delay-demand.

Step 3 : Output circuit graph G' after delay insertion and terminate.

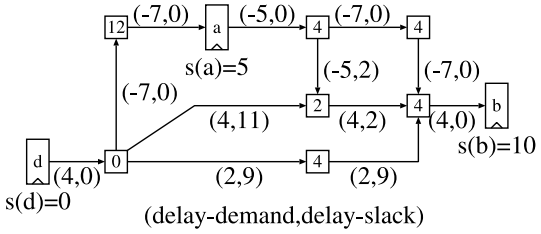


Fig. 8 Delay-slack and delay-demand of edges in G in Fig. 1.

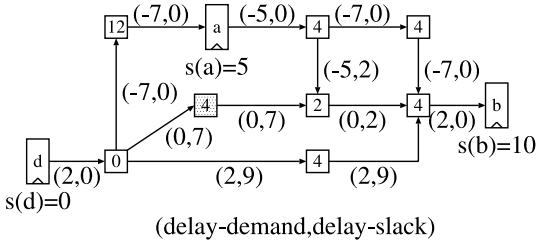


Fig. 9 Delay-slack and delay-demand after delay insertion.

For example, the proposed algorithm is applied to G in Fig. 1. Delay-slack and delay-demand are determined as shown in Fig. 8. Since a delay must be inserted where delay-demand is positive and can be inserted where delay-slack is positive, the algorithm inserts a delay as shown in Fig. 9. After delay insertion, since delay-slack and delay-demand are changed, the algorithm recalculates them. In Fig. 9, since the edges whose delay-demand are positive exist, the algorithm inserts delay more. Then, G' in Fig. 4 is obtained, and there is no edge with positive delay-demand. G' achieves the minimum clock period by delay insertion with less delay insertion than G'' in Fig. 5.

By the proposed algorithm, a delay is inserted to an edge whose delay-demand is positive. So we need to show that if there is a edge with positive delay-demand, then at least one edge whose delay-slack is positive exists among edges whose delay-demand are positive. Before proving the theorem, we show the following convenient definition and lemma.

Definition 4 (*critical path*): For a given clock schedule S that satisfies Setup constraints with clock period T, a path from register r_i to r_j is called a critical path, if the sum of edge weights is $S(r_j) - S(r_i) + T$.

If all the register is ticked at the same time and T is equal to the minimum clock period in complete-synchronous framework, a path whose sum of edge weights is the maximum delay between registers is a critical path. So this definition is an enhancement of the definition in complete-synchronous framework.

Lemma 1: When a clock schedule S with clock period T satisfies Setup constraints, an edge e belongs to a critical path if $slack(e) = 0$.

Proof. Assume that the paths that determine *complete* and *setup* of e are defined as in the proof of Theorem 3. In this

case, $slack(e)$ is

$$slack(e) = S(r_b) - S(r_a) + T - (d(P^I) + d(e) + d(P^O)).$$

So if $slack(e) = 0$, we have

$$d(P^I) + d(e) + d(P^O) = S(r_b) - S(r_a) + T.$$

Then, (P^I, e, P^O) is a critical path and e belongs to a critical path. □

Theorem 5: If there is a register pair that violates a Hold constraint in a clock schedule S that satisfies Setup constraints, there is at least one edge whose delay-slack is positive among edges whose delay-demands are positive.

Proof. Assume that (r_a, r_b) violates a Hold constraint in clock schedule S that satisfies Setup constraints with clock period T. Since (r_a, r_b) violates Hold constraint, we have

$$S(r_b) - S(r_a) > d_{min}(r_a, r_b). \quad (1)$$

Let $P = (e_1, e_2, \ldots, e_c)$ be a path that gives the minimum delay from register r_a to r_b. Since each element has a unique delay, we have

$$d_{min}(r_a, r_b) = d(P). \quad (2)$$

Assume that all edges on P have no delay-slack. From Lemma 1, e_i belongs to a critical path. Let P_i be the critical path from register r_i^I to r_i^O to which e_i belongs. Let P_i^I be the path from r_i^I to the previous edge of e_i on P_i and P_i^O be the path from the next edge of e_i to r_i^O on P_i (Fig. 10). Note that $d(P_i) = d(P_i^I) + d(e_i) + d(P_i^O)$.

Since P_i and P_{i+1} are critical paths, we have

$$S(r_i^O) - S(r_i^I) + T = d(P_i^I) + d(e_i) + d(P_i^O), \quad (3)$$
$$S(r_{i+1}^O) - S(r_{i+1}^I) + T = d(P_{i+1}^I) + d(e_{i+1}) + d(P_{i+1}^O). \quad (4)$$

Notice also that (P_{i+1}^I, P_i^O) is a path from r_{i+1}^I to r_i^O with weight $d(P_{i+1}^I)+d(P_i^O)$. Since Setup constraint of the register pair (r_{i+1}^I, r_i^O) is satisfied, we have

$$S(r_{i+1}^I) - S(r_i^O) \le T - (d(P_{i+1}^I) + d(P_i^O)). \quad (5)$$

From (3), (4), and (5), we have

$$S(r_{i+1}^O) - S(r_i^I) \le (d(P_i^I) + d(e_i) + d(e_{i+1}) + d(P_{i+1}^O)) - T.$$

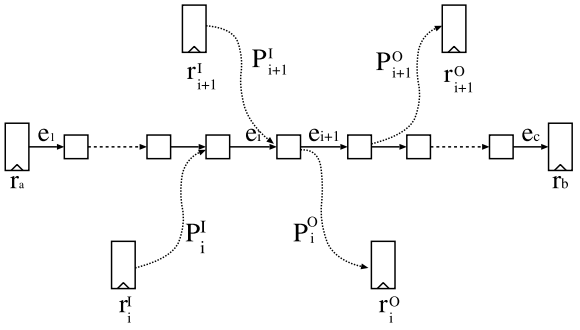


Fig. 10 A path P that consists of edges with no delay-slack.

Note that $(P_i^I, e_i, e_{i+1}, P_{i+1}^O)$ is a path from r_i^I to r_{i+1}^O.

Similarly we consider all edges on P. We have

$$S(r_c^O) - S(r_1^I) \le (d(P_1^I) + d(P) + d(P_c^O)) - T. \quad (6)$$

Since r_a is the head of e_1, $r_1^I = r_a$ and $d(P_1^I) = 0$. Since r_b is the tail of e_c, $r_c^O = r_b$ and $d(P_c^O) = 0$. Then inequality (6) is rewritten as

$$S(r_b) - S(r_a) \le d(P) - T.$$

Further this is rewritten by equality (2) as

$$\begin{aligned} S(r_b) - S(r_a) &\le d_{min}(r_a, r_b) - T \\ &< d_{min}(r_a, r_b). \end{aligned}$$

This contradicts the inequality (1). Therefore there is at least one edge whose delay-slack is positive among edges whose delay-demand are positive. □

By delay insertion, delay-slack and delay-demand of each edge never increase. If the delay is inserted to an edge e, then $slack(e) = 0$ or $demand(e) = 0$. So delay is never inserted to e again. Therefore, the delay insertion to edges is repeated at most the number of edges.

4.4 Computational Complexity

As the algorithm in [3], let n be the number of vertices in a constraint graph, m be that of directed edges in a constraint graph, g be that of vertices in a circuit graph, and p be that of directed edges in a circuit graph. In Step 1, the computation time for the maximum delay-to-register ratio is $O(mn)$ [7], the computation time for clock scheduling is also $O(mn)$. In Step 2, the computation times for *break*, *complete*, *setup*, *hold* are $O(g^2)$. It is repeated at most p times, so the computation time of this algorithm is $O(g^2 p)$.

5. Experiments

We implemented the algorithm in [3] and the proposed algorithm in a PC with a 3.06 GHz/512 K Intel Pentium-4 CPU and 512 MB RAM. We performed delay insertion on the LGSynth91 benchmark suite. In experiments, we assume that each gate has unit delay, and routing delays and register delays are zero. Moreover, inputs and outputs are regarded as one vertex in constraint graph so that the clock timings of inputs and outputs are set to same.

Among 24 circuits in benchmark suite, the maximum delay between registers is determined by a self-loop (a signal path from a register to the same register) in 8 circuits. In this case, the minimum clock period in semi-synchronous framework is equal to that in complete-synchronous framework. For the other 16 circuits, the minimum clock period in semi-synchronous framework is improved 17.4% from complete-synchronous framework on the average. Among improved 16 circuits, the minimum clock period by delay insertion is equal to the minimum clock period in 10 circuits. That is, the minimum clock period by delay insertion is achieved without delay insertion in 10 circuits, but

Table 1 Results.

Circuit	Num. of gates	$\|V_r\|$	$\|A_r\|$	clock period			Algorithm in [3]		Proposed algorithm	
				CS	SS (%)	Fin (%)	delay	Time [s]	delay (%)	Time [s] (%)
s298	119	15	84	9	6.00 (66.7)	5.34 (59.3)	71	0.17	14 (19.7)	0.08 (47.1)
s344	160	16	86	19	17.00 (89.5)	10.00 (52.6)	451	1.30	126 (27.9)	0.23 (17.7)
s349	161	16	86	19	17.00 (89.5)	10.00 (52.6)	451	1.14	126 (27.9)	0.24 (21.1)
s444	181	22	173	11	7.00 (63.6)	6.59 (59.9)	70	0.57	19 (27.1)	0.16 (28.1)
s526	193	22	165	9	6.00 (66.7)	5.50 (61.1)	66	0.38	12 (18.1)	0.18 (47.4)
s1423	657	75	1897	59	54.00 (91.5)	53.00 (89.1)	6172	217.72	3779 (61.2)	5.90 (2.7)
average	—	—	—	—	— (77.9)	— (62.6)	—	—	— (30.6)	— (27.3)

the minimum clock period can be improved by delay insertion in 6 circuits. For these 6 circuits, the minimum clock period in semi-synchronous framework is improved 22.1% from complete-synchronous framework on the average and improved 18.3% more by delay insertion on the average.

The results for the circuits that can be improved by delay insertion are shown in Table 1. In Table 1, the number of gates is shown by Num. of gates, the minimum clock period in complete-synchronous framework by CS, the minimum clock period in semi-synchronous framework by SS (the percentages in SS are the ratios of the minimum clock period in semi-synchronous framework to that in complete-synchronous framework), the minimum clock period by delay insertion by Fin (the percentages in Fin are the ratios of the minimum clock period by delay insertion to that in complete-synchronous framework), and the amount of inserted delays in previous algorithm and proposed algorithm by delay.

The amount of delay insertion is reduced by about 69.4% and computation time is reduced by about 72.7% compared to the algorithm in [3].

6. Conclusion and Future Works

In this paper, we propose a clock period minimization algorithm of semi-synchronous circuits by delay insertion and compare it with the algorithm in [3] in terms of the amount of delay insertion and computational complexity. We realize a lower bound of the clock period by less delay insertion than the algorithm in [3], because proposed algorithm inserts delay less than or equal to the delay-slack. The proposed algorithm is $O(\frac{np}{g})$ times faster than the algorithm in [3], and we show its efficiency by experiments.

As future works, we consider where delay is inserted and how the clock timing for each register is determined for the minimization of inserted delays and computation time. Finally, we want to propose a delay insertion algorithm for the real delay model (that is, maximum delay is not equal to minimum delay for each element).

References

[1] M. Saitoh, M. Azuma, and A. Takahashi, "A clustering based fast clock schedule algorithm for light clock-trees," IEICE Trans. Fundamentals, vol.E85-A, no.12, pp.2756–2763, Dec. 2002.

[2] S. Ishijima, T. Utsumi, T. Oto, and A. Takahashi, "A semi-synchronous circuit design method by clock tree modification," IEICE Trans. Fundamentals, vol.E85-A, no.12, pp.2596–2602, Dec. 2002.

[3] 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.

[4] T. Yasui, K. Kurokawa, M. Toyonaga, and A. Takahashi, "A circuit optimization method by the register path modification in consideration of the range of feasible clock timing," DA Symposium 2002, pp.259–264, 2002.

[5] J. Fishburn, "Clock skew optimization," IEEE Trans. Comput., vol.39, no.7, pp.945–951, 1990.

[6] A. Takahashi and Y. Kajitani, "Performance and reliability driven clock scheduling of sequential logic circuits," ASP-DAC'97, pp.37–43, 1997.

[7] R. Karp, "A characterization of minimum cycle mean in a digraph," Discrete Math., vol.23, pp.309–311, 1978.

Yukihide Kohira received his B.E. degree in electrical and electronic engineering from Tokyo Institute of Technology, Tokyo, Japan, in 2003. He is currently a M.E. student of Department of Communications and Integrated Systems in Tokyo Institute of Technology. He will receive his M.E. degree in 2005, and will be a D.E. student 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.