社団法人　情報処理学会　研究報告
IPSJ SIG Technical Reports

2005－ＳＬＤＭ－120（８）
2005／5／20

マルチクロックサイクルパスを含む準同期式回路ための最小クロック周期計算アルゴリズム

BakhtiarAffendi ROSDI[†]　　高橋　篤司[†]

† 東京工業大学大学院理工学研究科集積システム専攻
〒 152-8552 東京都目黒区大岡山 2-12-1
E-mail: †{fendi,atushi}@lab.ss.titech.ac.jp

あらまし　データが２クロックサイクル以上で伝播するレジスタ間パスはマルチクロックサイクルパスと呼ばれる。マルチクロックサイクルパスを含む回路では、シングルクロックサイクルパスだけからなる回路と違って、回路の動作可能なクロック周期に下限だけではなく、上限も存在する。シングルクロックサイクルパスだけからなる回路の最小クロック周期は２分探索等で求めることができるがそのようなアルゴリズムではマルチクロックサイクルパスを含む回路の最小クロック周期を求められない場合も存在する。そこで、本研究ではマルチクロックサイクルパスを含む回路の最小クロック周期を求めるアルゴリズムを提案する。
キーワード　マルチクロックサイクル，準同期式回路，最小クロック周期

An algorithm to calculate the minimum clock period of a semi-synchronous circuit that contains multi-clock cycle path

Bakhtiar AFFENDI ROSDI[†] and Atsushi TAKAHASHI[†]

† Department of Communications and Integrated Systems, Tokyo Institute of Technology
Ookayama 2-12-1, Meguro-ku, Tokyo, 152-8552 Japan
E-mail: †{fendi,atushi}@lab.ss.titech.ac.jp

Abstract　Multi-clock cycle path is a path from register to register where data transmission is more than one clock-period. For a circuit that contains multi-clock cycle path, there are lower bound and upper bound of the clock-period, which are different from a circuit that contains single-clock cycle path only. For a circuit contains single-clock cycle path only, the minimum feasible clock-period can be obtained by graph-theoretic approach with binary search and etc. If we use such approach to determine the minimum feasible clock-period of a circuit that contains multi-clock cycle path, there are some cases that cannot determine the minimum feasible clock-period. In this paper, we propose a new algorithm that determines the minimum feasible clock-period of a circuit that contains multi-clock cycle path.
Key words　Multi-clock cycle, semi-synchronous circuit, minimum clock-period

1. Introduction

A complete synchronous circuit is a circuit in which the clock is assumed to be distributed periodically and simultaneously to each individual register. The clock frequencies of complete synchronous circuits are decided based on the maximum delay time of paths between clocked registers under the assumption that all paths propagate signals within one clock cycle. The assumption is invalid in many circuits including multi-clock cycle paths on which signals can be propagated with 2 or more clock cycles.

While, due to the difficulty of clock-distribution and the increase of instantaneous power consumption and noise, the circuit design methodology without zero clock-skew is becoming popular. However, clock-schedule design methodology for a circuit that contains multi-clock cycle path is not established yet. In this paper, we propose an algorithm that determines the minimum feasible clock-period in terms of clock-scheduling of a circuit that contains multi-clock cycle path.

The minimum feasible clock-period in terms of clock-scheduling is obtained by linear programming [1], by graph-theoretic approach with binary search [2], [7], or by graph-theoretic approaches without binary search [3]. The method using graph-theoretic approach with binary search [7] is the most practical fast clock-schedule design algorithm that has been introduced. However, when we apply the algorithm [7] to a circuit that contains multi-clock cycle path, there are some cases that the algorithm cannot determine the minimum feasible clock-period.

For a circuit that contains multi-clock cycle path, there are lower bound and upper bound of the clock-period, which is different from a circuit that contains single-clock cycle path only. Therefore the range of feasible clock-period should be taken into account in clock-schedule design.

In this paper, first, we discuss on the constraint of a circuit that contains multi-clock cycle path. The constraints will take account the range of feasible clock-period in order to make the circuit tolerable to the clock jitter. Based on the constraints, we propose a new algorithm that can determine the minimum feasible clock-period of a circuit that contains multi-clock cycle path. Our proposed algorithm is based on the algorithm that has been introduced in [7]. Our algorithm uses the information from negative cycle in the constraint graph to find the lower and upper bound of feasible clock-period. By utilizing the value of bound, we will determine the minimum feasible clock-period by the binary search efficiently.

In experiments, we show that our proposed algorithm can determine the minimum feasible clock-period of a circuit that contains multi-clock cycle path efficiently, which cannot be determined by the algorithm shown in [7]. We had implemented the proposed algorithm to the reduction of the number of intermediate registers algorithm that has been introduced in [4]. In experiments to a 8-bit pipelined adder, we show that when we use the proposed algorithm, we can increase the reduction of the number of intermediate registers.

2. Preliminaries

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.

We assume the framework that a circuit works correctly if the following two types of constraints are satisfied for each register pair with signal propagation [1]:

Setup Constraint

$$s(u) - s(v) \leqq T - d_{\max}(u,v) \tag{1}$$

Hold Constraint

$$s(v) - s(u) \leqq d_{\min}(u,v) \tag{2}$$

where T is the clock-period and $d_{\max}(u,v)$ ($d_{\min}(u,v)$) is the maximum (minimum) propagation delay from register u to register v along a combinatorial circuit.

Multi-clock cycle paths are often used in a conventional zero clock-skew design in order to avoid the performance limitation caused by the maximum signal delay between registers. The constraints for a register pair with signal propagation defined as above are extended so that they correspond to general situations [4]:

Setup Constraint

$$s(u) - s(v) \leqq \beta_{u,v}T - d_{\max}(u,v) \tag{3}$$

Hold Constraint

$$s(v) - s(u) \leqq d_{\min}(u,v) - \alpha_{u,v}T \tag{4}$$

where $\beta_{u,v}$ and $\alpha_{u,v}$ are given constants ($\beta_{u,v} > \alpha_{u,v} \geqq 0$). Note that for a pair with single-clock cycle path, $\beta_{u,v}$ and $\alpha_{u,v}$ are given as 1 and 0, respectively. This formulation enables us to handle multi-clock cycle paths, mixture of positive-edge and negative-edge registers, latch based circuitry, and multi-clocks that have different periods, etc.

If $\alpha_{u,v}$ is 0 for every pair, the feasible clock-period has no upper bound. That is, if T is feasible then for any T' ($T' \geqq T$) is feasible. However, the feasible clock-period has an upper bound if $\alpha_{u,v}$ is not 0 for some pairs.

From the above constraints, when the clock schedule $s(u), s(v)$ and the signal propagation delay d_{min}, d_{max} are known, the minimum and maximum feasible clock-period T_{min} and T_{max} can be determined by setup and hold constraint, respectively.

If the clock timings $s(u), s(v)$ can be scheduled, T_{min} and T_{max} are dependent on each other. Therefore, to get a circuit that is tolerable to the clock jitter and etc., the circuit needs to work correctly within clock period range T_{min} and T_{max}. Thus, we have to minimize T_{min} under the constraint that the circuit is feasible at clock period range T_{min} and T_{max}. Let $T_{max} - T_{min} = \delta$, then the above constraints will be:

Setup Constraint

$$s(u) - s(v) \leqq \beta_{u,v}T_{min} - d_{\max}(u,v) \tag{5}$$

Hold Constraint

$$s(v) - s(u) \leqq d_{\min}(u,v) - \alpha_{u,v}T_{min} - \alpha_{u,v}\delta \tag{6}$$

where δ is the clock-period range. Therefore if the clock-period range δ is given, by using the above constraints, clock timings $s(u), s(v)$ can be determined, so that the circuit works correctly within T_{min} and $T_{min} + \delta$.

In this paper, our target is to get a circuit that works correctly at target clock period range δ. By replacing $d_{min}(u,v)$ value with $d_{min}(u,v) - \alpha_{u,v}\delta$ at constraint (4), we can get a circuit that tolerable at clock period range δ. Therefore, in this paper, we consider constraints (3) and (4).

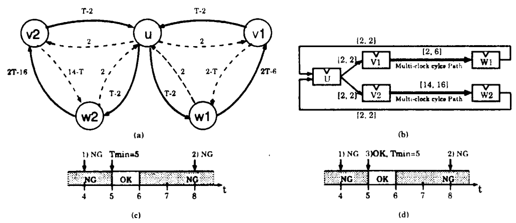

Fig. 1 (a) Constraint Graph G^1.
(b) Circuit Graph CG^1 (Contains multi-clock cycle path).
(c) Min Clock Period Calculation by the algorithm shown in [7].
(d) Min Clock Period Calculation by proposed algorithm.

These constraints are represented by the constraint graph. The constraint graph $G(V, E)$ of a circuit is defined as follows: a vertex $v \in V$ corresponds to a register; a directed edge $(u, v) \in E$ corresponds to either type of constraints; edge (u, v) corresponding to the setup (hold) constraint is called setup-edge (hold-edge), and the weight $w(u, v)$ of (u, v) is $\beta_{u,v}T - d_{\max}(u, v)(d_{\min}(u, v) - \alpha_{u,v}T)$. It is known that the constraints can be satisfied if and only if the constraint graph contains no negative weight cycle [3]. The constraint graph G when the clock-period is t is denoted by G_t.

In a constraint graph, cycle weight $w(C)$ is the sum of edge weights of a cycle C. A negative cycle is a cycle where the cycle weight is negative.

[Definition 1] Cycle coefficient of C is defined as k when $w(C) = kT + w$, where T is the clock-period and k and w be constants. A cycle where the cycle coefficient k is 0, greater than 0 and less than 0, is said to be **0-type**, **P-type** and **M-type**, respectively.

[Definition 2] For a cycle C where $w(C) = kT + w$ in the constraint graph, if C is **P-type**, *Bound*$(C) = w/k$. If C is **M-type**, *Bound*$(C) = -w/k$, while, if C is **0-type**, *Bound*$(C) = \infty$.

Example: For constraint graph G^1 shown in Fig. 1 (a), cycle $C_1 = (u, w2)(w2, v2)(v2, u)$ $(w(C_1) = 4T - 20)$ is **P-type**, *Bound*$(C_1) = 5$. Cycle $C_2 = (u, v1)(v1, w1)(w1, u)$ $(w(C_2) = -T + 6)$ is **M-type**, *Bound*$(C_2) = 6$.

[Theorem 1] Let C be a negative cycle in a constraint graph G_t. If C is **0-type**, then for any t', there are negative cycles in the constraint graph $G_{t'}$.

[Theorem 2] Let C be a cycle in a constraint graph G. If C is **P-type**, then for any $t <$ *Bound*(C), there are negative cycles in the constraint graph G_t. While, if C is **M-type**, then for any $t >$ *Bound*(C), there are negative cycles in the constraint graph G_t.

If the circuit is infeasible at given clock-period t, then a negative cycle C in constraint graph G_t is found. From above theorems, it is clear that, if the type of C is **0-type**, the circuit is infeasible for any clock-period t. While, if the type of C is **P-type** (**M-type**), the *Bound*(C), which is larger (smaller) than t, gives a lower (upper) bound of the feasible clock-period.

```
Procedure MinClock( $G(V, E)$ )
Input:  constraint graph $G(V, E_{hold} \cup E_{setup})$.
Output:  minimum feasible clock period $T$.
 (1)  /* Compute initial lower and upper bounds */
 (2)  $L_{\mathrm{self}} := \max_{(u,u)\in E_{hold}} d_{max}(u, u)$.
 (3)  $L_{\mathrm{diff}} := \max_{(u,v)\in E_{hold}} (d_{max}(u, v) - d_{min}(u, v))$.
 (4)  $L := \max\{L_{\mathrm{self}}, L_{\mathrm{diff}}\}$.
 (5)  $U := \max_{(u,v)\in E_{hold}} d_{max}(u, v)$.
 (6)  /* Check lower bound $L$ */
 (7)  if IsNoNegativeCycle($G, L$ ) = "Yes " return $L$.
 (8)  /* Check upper bound $U$ */
 (9)  if IsNoNegativeCycle($G, U$ ) = "No " return $\infty$.
(10)  /* Binary Search */
(11)  while $(U - L > \varepsilon)$ do
(12)      $M := (U + L)/2$.
(13)      if IsNoNegativeCycle ($G, M$) = "Yes "
(14)      then $U := M$,
(15)      else $L := M$ endif
(16)  endwhile
(17)  $T := U$.
(18)  return $T$.
```

Fig. 2 Minimum feasible clock period algorithm of the circuit that contains single-clock cycle path [7].

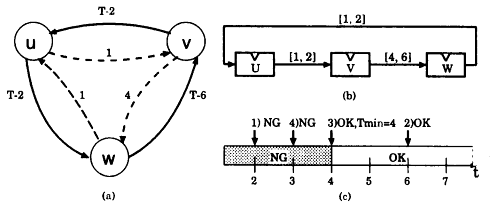

Fig. 3 (a) Constraint Graph G^2.
(b) Circuit Graph CG^2 (Consists only single-clock cycle path).
(c) Min Clock Period Calculation by the algorithm shown in [7].

Note that, in a constraint graph of a circuit that contains single-clock cycle path only, there are two types of cycle that are **P-type** and **0-type**, while in a constraint graph of a circuit that contains multi-clock cycle path, there are three types of cycle that are **P-type**, **M-type** and **0-type**.

3. Minimum Feasible Clock Period

3.1 Circuit contains single-clock cycle path only

There is no maximum feasible clock-period of a circuit that contains single-clock cycle path only. That is, if T_{min} is the minimum feasible clock-period, then for any t $(t \geqq T_{min})$ is feasible.

The minimum feasible clock-period of a circuit that contains single-clock cycle path only can be determined by graph-theoretic approach with binary search [7] as shown in Fig. 2. In the algorithm shown above, the maximum signal propagation delay from register to the same register gives a lower bound of feasible clock-period. The difference of the maximum and minimum signal propagation delay from register to another register gives also a lower bound of feasible clock-period. They adopt the larger of these two lower bounds as an initial lower bound L of the binary search. They adopt the maximum signal propagation delay between registers as an initial upper bound U since it gives a feasible clock-period even in zero clock-skew framework.

For the first step of the algorithm, the initial lower bound L and upper bound U will be checked. If L is feasible the algorithm is stopped and output L as the minimum feasible clock-period. While, if L is infeasible, U will be checked to confirm there are no **0-type** negative cycles. If there are **0-type** negative cycle, the circuit is infeasible and the algorithm is stopped. If U is feasible, the algorithm does binary search by adjusting the lower and upper bounds to determine the minimum feasible clock-period.

By using the algorithm shown in [7] let us determine the minimum feasible clock-period of the circuit shown in Fig. 3 (b). Note that, throughout this paper, the precision ε used is 1. In this example, initial lower bound $L = 2$ and initial upper bound $U = 6$. So, the algorithm does binary search between 2 and 6 as follows:

Check L: When $L = 2$ in G^2, cycle $C_1 = (u,w)(w,v)(v,u)$ $(w(C_1) = 3T - 10)$ is negative, so next step is check the initial upper bound.

Check U: When $U = 6$ in G^2, there is no negative cycle, so next step is check the half value ((U+L)/2).

Check M: When $M = 4$ $((6+2)/2)$ in G^2, there is no negative cycle, so 4 will be new upper bound U.

Check M: When $M = 3$ $((4+2)/2)$ in G^2, cycle $C_1 = (u,w)(w,v)(v,u)$ $(w(C_1) = 3T - 10)$ is negative. Since $U - L = 1$, return 4 as the minimum feasible clock-period T. The flow when we apply the algorithm is shown in Fig. 3 (c).

For a circuit that contains single clock cycle path only, if the circuit is feasible then the circuit is feasible at the initial upper bound U, otherwise the circuit is infeasible. However, for a circuit that contains multi-clock cycle path, although the circuit is infeasible at initial upper bound U there are some possibilities that the circuit is feasible at clock-period t $(t < U, t > U)$. When the initial upper bound U is infeasible, the algorithm [7] assumes that the circuit is infeasible for any clock-period and stop. As we said earlier, the assumption is correct for a circuit that contains single clock cycle path only, while for the circuit that contains multi-clock cycle path the assumption is incorrect. Therefore, the above approach might miss the minimum feasible clock-period. For example, let us determine the minimum feasible clock-period of the circuit shown in Fig. 1 (b). Initial lower bound $L = 4$ and initial upper bound $U = 8$. So, the algorithm does binary search between 4 and 8 as follows:

Check L: When $L = 4$ in G^1, cycle $C_1 = (u,w2)(w2,v2)(v2,u)$ $(w(C_1) = 4T-20)$ is negative, so next step is check the initial upper bound.

Check U: When $U = 8$ in G^1, cycle $C_2 = (u,v1)(v1,w1)(w1,u)$ $(w(C_2) = -T + 6)$ is negative and the algorithm is stopped.

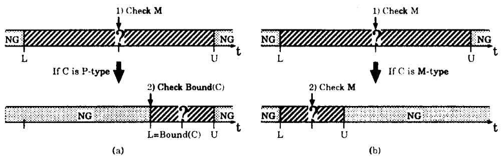

Fig. 4 (a) If C is **P-type** then $L = Bound(C)$ and check L.
(b) If C is **M-type** then $U = Bound(C)$.

The flow when we apply the algorithm is shown in Fig. 1 (c). As you can see from the above example, the algorithm [7] is stopped after checking the initial upper bound U and assumes that the circuit is infeasible at any clock period. The algorithm cannot determine the minimum feasible clock-period which is 5.

3.2 Circuit contains multi-clock cycle path

There is maximum feasible clock-period of a circuit that contains multi-clock cycle path. That is, if T_{min} and T_{max} is the minimum and maximum feasible clock-period, respectively, then for t $(t < T_{min}, t > T_{max})$ is infeasible.

Our proposed algorithm does binary search between lower and upper bounds same as in the algorithm shown in [7]. We extend the algorithm [7] by introducing checking the type of cycle when we detected the negative cycle in the constraint graph. If the circuit is infeasible at given clock-period, a negative cycle will be found. Based on the type of negative cycle and the *Bound* value, our algorithm will adjust the lower and upper bounds and determines the minimum feasible clock-period.

For the initial value of lower bound L and upper bound U of the binary search, we adopt the same approach as in the algorithm shown in [7]. Initial lower bound L will be checked whether it is feasible or not, if L is feasible, then L is the minimum feasible clock period. Otherwise, a negative cycle C is found. If C is **0-type** or **M-type**, the circuit is infeasible. While, if C is **P-type** then check an initial upper bound U whether it is feasible or not.

If the initial upper bound U is feasible, then the algorithm does binary search to determine the minimum feasible clock period. Otherwise, a negative cycle C is found. In case, C is **0-type**, the circuit is infeasible. In case C is **P-type**, *Bound*(C) is our new lower bound L and check whether L is feasible or not. If our new lower bound L is feasible then L is the minimum feasible clock period, otherwise repeat until L is feasible or type of C is **0-type** or **M-type**, where the circuit is infeasible. In case C is **M-type**, *Bound*(C) is our new upper bound U and check whether $U < L$ or not. If $U < L$, then the circuit is infeasible, otherwise, our algorithm does binary search by adjusting the lower and upper bounds to determine the minimum feasible clock period.

In binary search, the algorithm will check whether the constraint graph $G_M(M = (U + L)/2)$ contains any negative cycle or not. If there are no negative cycles in G_M, then M will be our new upper bound U and continue do binary

```
Procedure MinClock( $G(V, E)$ )
Input: constraint graph $G(V, E_{hold} \cup E_{setup})$, target clock period range $\delta$.
Output: minimum feasible clock period $T$.
( 1 ) /* Compute initial lower and upper bounds */
( 2 ) $L_{\text{self}} := \max_{(u,u) \in E_{hold}} d_{max}(u, u)$.
( 3 ) $L_{\text{diff}} := \max_{(u,v) \in E_{hold}} (d_{max}(u, v) - d_{min}(u, v))$.
( 4 ) $L := \max\{L_{\text{self}}, L_{\text{diff}}\}$.
( 5 ) $U := \max_{(u,v) \in E_{hold}} \{(d_{max}(u, v) + s(u) - s(v))/\beta_{u,v}\}$.
( 6 ) /* Check lower bound $L$ */
( 7 ) if IsNoNegativeCycle($G, L$) = "Yes " return $L$,
( 8 ) else
( 9 )   if ($C$ is 0-type or M-type) return $\infty$.
(10) endif
(11) /* Check upper bound $U$ */
(12) if IsNoNegativeCycle($G, U$) = "No " then
(13)   if ($C$ is 0-type) return $\infty$.
(14)   else if ($C$ is P-type) then
(15)     $L$ := Bound($C$).
(16)     while (IsNoNegativeCycle($G, L$) = "No ") do
(17)       if ($C$ is 0-type or M-type) return $\infty$.
(18)       $L$ := Bound($C$).
(19)       if IsNoNegativeCycle ($G, L$) = "Yes "
(20)         return $L$.
(21)     endwhile
(22)   else /* M-type */
(23)     $U$ := Bound($C$).
(24)   endif
(25) endif
(26) /* Binary Search */
(27) while ($U - L > \varepsilon$) do
(28)   $M := (U + L)/2$.
(29)   if IsNoNegativeCycle ($G, M$) = "Yes " then
(30)     $U := M$,
(31)   else
(32)     if ($C$ is 0-type) return $\infty$.
(33)     else if ($C$ is P-type)
(34)       $L$ := Bound($C$).
(35)       if IsNoNegativeCycle($G, L$) = "Yes "
(36)         return $L$,
(37)       else
(38)         if ($C$ is 0-type or M-type) return $\infty$.
(39)       endif
(40)     else /* M-type */
(41)       $U$ := Bound($C$).
(42)     endif
(43)   endif
(44) endwhile
(45) $T := U$.
(46) return $T$.
```

Fig. 5 Minimum feasible clock period algorithm of the circuit that contains multi-clock cycle path.

search. Otherwise, if there are negative cycles in G_M then the algorithm will find a negative cycle C and check the type of it. In case C is **0-type**, from Theorem 1, the circuit is infeasible for any clock-period. From Theorem 2, in case C is **P-type** then *Bound(C)* is our new lower bound L and check whether L is feasible or not (Refer Fig. 4 (a)). If our new lower bound L is feasible then L is the minimum feasible clock period, otherwise, continue do binary search. In case C is **M-type** then *Bound(C)* is our new upper bound U (Refer Fig. 4 (b)), and continue do binary search.

The new algorithm to determine the minimum feasible clock period of a circuit that contains multi-clock cycle path is shown in Fig. 5. By using the proposed algorithm, let us determine the minimum feasible clock period of the circuit shown in Fig. 1 (b). Our target clock period range δ is 0. Initial lower bound $L = 4$ and initial upper bound $U = 8$. So, the algorithm does binary search between 4 and 8 as follows:
Check L: When $L = 4$ in G^1, cycle $C_1 = (u, w2)(w2, v2)(v2, u)$ ($w(C_1) = 4T - 20$) is negative.
Check U: When $U = 8$ in G^1, cycle $C_2 = (u, v1)(v1, w1)(w1, u)$ ($w(C_2) = -T + 6$) is negative, *Bound*(C_2) = 6, therefore 6 is our new upper bound.
Check M: When $M = 5$ ((6+4)/2) in G^1, no more negative cycle and $U - L = 1$, so the minimum feasible clock period is 5.

The flow when we apply the algorithm is shown in Fig. 1 (d). The algorithm can determine the minimum feasible clock period of the circuit which is 5.

4. Experiment

The algorithm proposed are written in C++ and implemented on Pentium 4 (CPU 3Ghz, memory 513764Kb). For comparisons, the algorithm shown in [7] is executed on the same machine.

The data used in experiments are shown in Table 1. Among circuits, block1, 2, 3, 4 and 5 are extracted from circuits designed in an industry in 0.25[μm] (block1, 2, 3) and in 0.18[μm] (block4 and 5) technologies. While, 8bitadd1 and 2 are parallel pipelined 8 bit adder using 0.35[μm] technology. Among circuits, block1, 2, 3, 8bitadd1 and 2 contain multi-clock cycle paths. Feasible clock-period of block2, 3, 8bitadd1 and 2 have an upper bound, since they contain a register pair with non-zero $\alpha_{u,v}$. In experiments, the timing of each I/O pin is scheduled as well as registers. Target clock period range $\delta = 0$.

In Table 2, the comparisons on computation times and number of repetition of bellman-ford and negative cycle detection (BF-ncd) algorithm between our proposed algorithm and original algorithm [7] are shown. The computation times of two algorithms are almost same if the number of repetition is same. This shows that the overhead of computation of upper (lower) bound of the feasible clock period is small enough. As shown in circuit block5 and 8bitadd2, our proposed algorithm can reduced the repetition number of bellman-ford and negative cycle detection algorithm. There are slight improvement on the computation times of our proposed algorithm due to the reduction on the number of repetition.

In Table 3, it is shown that our proposed algorithm can determine the minimum (maximum) feasible clock period of a circuit that contains multi-clock cycle path which cannot be determined by the original algorithm [7].

The proposed algorithm is applied to the algorithm to re-

Table 1 Statistics of circuits.

circuit	*#reg*	*#path*	max-d[ps]	$T_{min}[ps]$	$T_{max}[ps]$
block1	1654	11697	11569	8323	∞
block2	6439	113101	11911	11654	11654
block3	7973	104136	11808	9553	12922
block4	7052	126559	8354	6256	∞
block5	12460	947082	12621	9665	∞
8bitadd1	54	126	2790	2407	2407
8bitadd2	62	136	2890	2231	2231

#reg : number of registers (including I/O pins).
#path : number of register pairs with signal paths.
max-d : maximum of maximum delay over clock-cycle.
T_{min} (T_{max}) : minimum (maximum) feasible clock-period.

Table 2 Comparison on the Number of repetition of BF-ncd.

			Proposed		Original	
circuit	$T_{min}[ps]$	$T_{max}[ps]$	*time[s]*	λ	time[s]	λ
block1	8323	∞	0.05	1	0.04	1
block2	11654	11654	0.95	1	0.92	1
block3	9553	12922	2.06	1	2.14	1
block4	6256	∞	1.50	1	1.56	1
block5	9665	∞	31.55	8	37.91	14
8bitadd1	2407	2407	0.00	4	NA	NA
8bitadd2	2231	2231	0.00	4	0.00	12

time : time for computation.
NA : Not Available.
λ : number of repetition of BF-ncd.

Table 3 Comparison on the Solution Determination.

		Proposed		Original	
$\delta[ps]$	circuit	$T_{min}[ps]$	$T_{max}[ps]$	$T_{min}[ps]$	$T_{max}[ps]$
150	8bitadd1	2560	2710	NA	NA
250	8bitadd2	2560	2810	OT	OT

NA : Not Available. OT : Out of Time.
δ : Target clock period range.

Table 4 Reduction on the number of intermediate registers.

		Proposed			Original [4]		
δ	$[T_{min}, T_{max}]$	*#reg.*	*#Int.*	Area	*#reg.*	*#Int.*	Area
(ps)	(ps)			(%)			(%)
	[1710, Inf.]	41	16	100	41	16	100
150	[2560, 2710]	**29**	**4**	**83**	**36**	**11**	**93**
200	[2560, 2760]	**37**	**12**	**95**	**38**	**13**	**96**
250	[2560, 2810]	**37**	**12**	**95**	**38**	**13**	**96**

#reg : number of registers (excluding I/O pins).
#Int. : number of intermediate registers.
Area : Circuit area.

duce the number of intermediate registers shown in [4]. In experiments, the timing of each I/O pin is fixed to 0, while the timing of registers is scheduled. In Table 4, it is shown that our proposed algorithm can removed more intermediate registers compared to the original algorithm [4].

5. Conclusions

This paper introduces a new algorithm that can determine the minimum feasible clock period of a semi-synchronous circuit that contains multi-clock cycle path. The result shown that it can determine the minimum feasible clock period of a circuit that contains multi-clock cycle path, which cannot be determined by the algorithm shown in [7]. When we applied the proposed algorithm to reduce the number of intermediate registers, it is shown that our proposed algorithm can removed more intermediate registers compared to the original algorithm shown in [4].

6. Acknowledgement

This work is supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc., Cadence Design Systems, Inc., Rohm Corporation and Toppan Printing Corporation.

References

[1] J. P. Fishburn: "Clock skew optimization", IEEE Trans. on Computers, **39**, 7, pp. 945–951 (1990).

[2] R. B. Deokar and S. S. Sapatnekar: "A graph-theoretic approach to clock skew optimization", Proc. International Symposium on Circuits and Systems (ISCAS), Vol. 1, pp. 407–410 (1994).

[3] A. Takahashi and Y. Kajitani: "Performance and reliability driven clock scheduling of sequential logic circuits", Proc. Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 37–42 (1997).

[4] B. A. Rosdi and A. Takahashi: "Reduction on the usage of intermediate registers for pipelined circuits", Proc. the Workshop on Synthesis and System Integration of Mixed Technologies (SASIMI 2004), pp. 333–338 (2004).

[5] E. L. Lawler: "Combinatorial Optimization, Networks and Matroids", Holt, Rinehart and Winston, New York (1976).

[6] R. B. Deokar and S. S. Sapatnekar: "A fresh look at retiming via clock skew optimization", Proc. 32nd DAC, pp. 310-315 (1995).

[7] A. Takahashi: "Practical Fast Clock Schedule Design Algorithms", 第 18 回 回路とシステム軽井沢ワークショップ論文集, pp.515-520, 2005 年 4 月 26 日.