論文 / 著書情報
Article / Book Information

論題(和文)	遅延挿入量最小化のためのクロックスケジューリングと遅延挿入手法
Title(English)	Clock Scheduling Method and Delay Insertion Method for Minimization of Inserted Delay
著者(和文)	小平行秀, 谷修平, 高橋篤司
Authors(English)	Yukihide Kohira, Shiyuuhei Tani, Atsushi Takahashi
出典(和文)	第21回 回路とシステム軽井沢ワークショップ 論文集, Vol. , No. , pp. 629-634
Citation(English)	Proc. the 21st Workshop on Circuits and Systems in Karuizawa, Vol. , No. , pp. 629-634
発行日 / Pub. date	2008, 4
URL	http://search.ieice.org/
権利情報 / Copyright	本著作物の著作権は電子情報通信学会に帰属します。 Copyright (c) 2008 Institute of Electronics, Information and Communication Engineers.

第21回　回路とシステム
軽井沢ワークショップ
The 21st Workshop on Circuits and Systems
in Karuizawa, April 21-22, 2008

遅延挿入量最小化のためのクロックスケジューリングと遅延挿入手法
Clock Scheduling Method and Delay Insertion Method for Minimization of Inserted Delay

小平 行秀 †, 谷 修平 †, 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

Yukihide KOHIRA†, Shuhei TANI†, Atsushi TAKAHASHI†
† Dept. of Communications and Integrated Systems, Tokyo Institute of Technology

1　はじめに

クロックを各記憶素子に同一周期ではあるが，同時に分配することを前提としていない一般同期方式に関する初期の研究では，与えられた論理回路に対して，各記憶素子へのクロック入力タイミングの決定法 [3, 2, 10, 9] や，クロック回路の設計法 [4, 5] に重点が置かれていた．しかし，入力として与えられる論理回路は，クロックを各記憶素子に同時に分配することを前提としている完全同期方式を前提に合成されており，一般同期方式の下で性能が改善できる回路とは限らない．クロック周期の短縮のために，完全同期方式では記憶素子間の遅延の最大値が減少するように論理合成をするが，一般同期方式では必ずしも記憶素子間の遅延の最大値が減少する必要はない．そのため，一般同期方式を前提とした論理回路の合成をすることで，高速化に伴う回路面積の増大などのコストを抑えた論理回路が得られる可能性が高い．

本稿では，各レジスタのクロック入力タイミングを自由に設定できるという仮定の下で，ゲートレベルにおいて，遅延挿入により所望のクロック周期を実現する論理回路修正手法を検討する．与えられた論理回路からの修正量を少なくするため，できるだけ少ない遅延挿入量で，所望のクロック周期を実現する論理回路を得ることを目指す．実際の回路での遅延の挿入は，短時間で演算結果を出力する大きいモジュールから演算結果の出力により長い時間がかかる小さいモジュールへの変更，より細い配線への変更などが考えられる．つまり，従来の完全同期方式では，クロック周期と回路面積はトレードオフの関係にあり，どちらか一方を小さくすると，もう一方は大きくなってしまう傾向にあるが，一般同期方式では，より小さい面積で，より小さいクロック周期を達成するような回路が得られる可能性がある．

一般同期方式における遅延挿入手法として，文献 [11, 7] では，数理計画法を用いて，クロック周期や遅延挿入量を最小化する手法が提案されているが，実際の回路に適用する場合，整数計画法に定式化しなければならず，大規模な回路では適用できない．文献 [12] では，各ゲートの最大遅延値と最小遅延値が等しい場合，最小クロック周期の下界を達成する手法が提案されている．しかし，遅延挿入量を決定する際に回路全体を探索するので計算時間が長く，最小クロック周期が増加しない最大の遅延量を挿入しているので遅延挿入量が多い．文献 [6] では，各ゲートの最大遅延値と最小遅延値が等しい場合，一部の制約を考慮せず各レジスタのクロック入力タイミングを決定し，その制約を違反する部分回路に対して，決定したクロック入力タイミングを満たすように遅延を挿入することで，最小クロック周期の下界を達成する手法が提案されている．この手法では，回路の構造に関係なく決定した遅延挿入箇所に，与えられたクロック入力タイミングの制約を満たすのに必要となるできるだけ少ない遅延値を挿入しているため，文献 [12] で提案されている手法よりは遅延挿入量が少ないが，クロック入力タイミングの決定法，遅延挿入箇所の決定法が最適ではなく，遅延挿入量が多い．

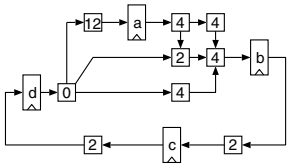


図 1: 回路 G.

そこで本稿では，文献 [6] で提案されている遅延挿入手法をベースにして，各ゲートの最大遅延値と最小遅延値が等しい場合に遅延挿入量の最小化を目指したクロック入力タイミングを決定する発見的手法と，遅延挿入箇所と量を決定する発見的手法を提案する．また，計算機実験において，提案手法は既存手法より遅延挿入量と計算時間を大幅に改善し，多くの回路で挿入する遅延量が最小となることを示す．

2　準備

本稿では，回路をグラフ $G = (V_g, E_g)$ として表現する．V_g はグラフの点集合で，入出力ピン，レジスタ，ゲート，配線に対応する．E_g はグラフの有向枝の集合で，信号伝搬に対応する．本稿では，各遅延素子は一意の非負の遅延値を有していると仮定する．ある点 $v \in V_g$ の遅延値を $d(v)$ と表記する．

回路のレジスタの集合を V_r とする．図 1 に回路の一例を示す．図 1 では，$\{a, b, c, d\}$ がレジスタの集合であり，レジスタ以外の点内の数字は遅延値を表す．レジスタの遅延値も点の重みとして表現できるが，本稿では説明を簡略にするため，レジスタの遅延値を 0 とする．

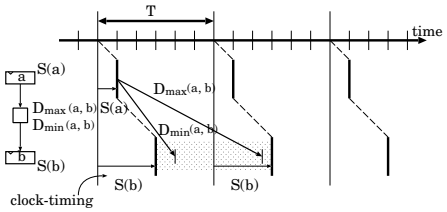


図 2: 同期回路のタイミング制約.

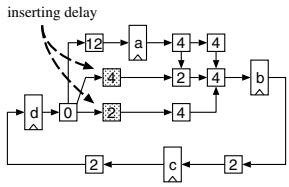


図 3: 回路 G に遅延を挿入した回路 G'.

2.1 一般同期式回路

クロック同期回路が正常に動作するための条件は，信号が伝搬する全てのレジスタ対が以下の 2 式を満たすことであると知られている [3] (図 2).

Setup 制約: 0 クロック制約

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

Hold 制約: 二重クロック制約

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

ここで，T はクロック周期，$d_{\max}(a, b)$，$d_{\min}(a, b)$ はレジスタ a, b 間の最大遅延，最小遅延，$S(a)$，$S(b)$ はレジスタ a，b のクロックタイミングを表す.

完全同期方式では，全てのレジスタに同じタイミングで同一周期のクロックが与えられることを前提とするため，クロック周期がレジスタ間の最大遅延未満の時，回路は正常に動作しないと判断される．一方，一般同期方式では，クロック周期がレジスタ間の最大遅延値未満で完全同期方式では正常に動作しないとみなされる回路でも，全ての制約を満たすのであれば回路は正常に動作すると正確に判断する.

本稿では，一般同期方式において，各レジスタに任意のクロックタイミングを設定できると仮定したときの回路 G の最小クロック周期を $T_S(G)$ とする．一般同期方式における回路 G の最小クロック周期 $T_S(G)$ は，回路 G から得られる制約グラフ $H(V_r, E_r)$ によって定められる [10]．制約グラフ $H(V_r, E_r)$ の点集合 V_r は回路のレジスタの集合に対応する．制約グラフの有向枝集合 E_r は 2 つの制約式に対応する．レジスタ a から b への重み $d_{\min}(a, b)$ の有向枝は Hold 制約に対応し，D 枝と呼ぶ．レジスタ b から a への重み $T - d_{\max}(a, b)$ の有向枝は Setup 制約に対応し，Z 枝と呼ぶ．以後，制約グラフ $H(V_r, E_r)$ を簡単に $H(G)$ と表記する．また，制約グラフの Z 枝はクロック周期 T の関数である．クロック周期 $T = t$ とした制約グラフを $H(G, t)$ と表記する.

定理 1 ([10]) 一般同期方式の最小クロック周期 $T_S(G)$ は，制約グラフ $H(G, t)$ が負閉路を持たない最小の t である.

すなわち，一般同期方式では，制約グラフ $H(G, T_S(G))$ において 0 閉路であり，制約グラフ $H(G, t)$ $(t < T_S(G))$ において負閉路である閉路が最小クロック周期を決めている.

2.2 遅延挿入

回路における遅延の挿入は，回路グラフにおいて 1 つの有向枝を，直列な 2 つの有向枝とその間に正の重みを持つ点に置き換えることに対応する．図 1 に示した回路グラフ G に遅延を挿入した回路グラフ G' を図 3 に示す.

本稿では，ゲートレベルの回路において，任意のクロックスケジュールが設定でき，各素子が一意の遅延値を有しているという仮定の下で，与えられた論理回路の修正量を少なくするため，できるだけ少ない遅延挿入量で一般同期方式で所望のクロック周期を実現する手法を提案する．所望のクロック周期が与えられた論理回路の最小クロック周期以上の場合，遅延挿入しなくても所望のクロック周期が達成できる．したがって，所望のクロック周期は与えられた論理回路の最小クロック周期未満の値が与えられるとする.

あるクロック周期で動作しない回路を動作するように修正するためには，制約グラフにある全ての負閉路を解消すれば良い．負閉路を解消するためには，負閉路上の枝重みを増加させれば良い．ここで，D 枝の枝重みは $D_{\min}$ であり，Z 枝の枝重みは $T - D_{\max}$ であるので，制約グラフの枝重みを増加させるには，レジスタ間の最小遅延を増加させるか，レジスタ間の最大遅延を減少させれば良い．本稿では，回路修正として遅延挿入のみを考慮するので，レジスタ間の最大遅延を減少させることができない．一般同期方式において遅延の挿入できる量の制約がなく，遅延値が減少しない場合，最小クロック周期の下界は**限界最小クロック周期**になることが知られている [12].

定義 1 ([12]) 限界最小クロック周期 $T_L(G)$

$$T_L(G) = \max_{L \in cycle\ set} \frac{D(L)}{N(L)}$$

ここで，$D(L)$ は G 中の閉路 L 上の遅延値の和，$N(L)$ は G 中の閉路 L 上にあるレジスタ数である.

限界最小クロック周期 $T_L(G)$ は Z 枝のみからなる制約グラフ $H_Z(G, t)$ が負閉路を持たない最小の t であることが知られている [6].

所望のクロック周期が限界最小クロック周期未満の場合，どのような遅延挿入しても所望のクロック周期が達成できない．したがって，所望のクロック周期は限界最小クロック周期以上の値が与えられるとする.

2.3 既存手法

文献 [6] では，一般同期方式において，遅延挿入によりクロック周期を最小化する手法が提案されている (図 4).

各レジスタにクロックタイミングが与えられた時に，各枝 e に対して，Setup 制約を満たしたままで挿入できる遅延量である**遅延余裕量** $slack(e)$，Hold 制約を満たすために挿入しなければならない遅延量である**遅延要求量** $demand(e)$ が定義される．この遅延量は，最も早く

入力： 回路 G

出力： 遅延挿入後の回路 G'

Step 1： $T_S(G)$ と $T_L(G)$ を求める．$T_S = T_L$ の場合は元の回路を出力し，終了．

Step 2： $T = T_L(G)$ において，Setup 制約のみを満たすクロックスケジュールを Z 枝のみからなる制約グラフの最短パスにより定める．

Step 3： $demand(e) > 0$ となる枝 e が存在しなくなるまで，以下の操作を繰り返す．

> 違反レジスタから幅優先探索し，$demand(e) > 0$ かつ $slack(e) > 0$ となる最初の枝 e に $\min\{slack(e), demand(e)\}$ の遅延を挿入する．遅延の挿入後に $slack$, $demand$ を再計算する．

Step 4： 遅延挿入後の回路グラフを G' として出力し，終了．

図 4: 既存手法 [6].

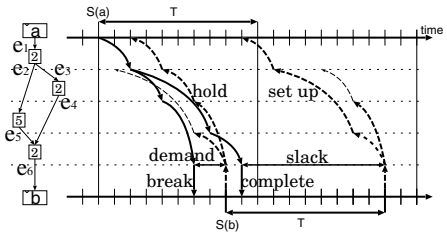


図 5: 遅延余裕量 $slack$ と遅延要求量 $demand$.

信号が到着する時間 $break(e)$，最も早く正確な信号に確定する時間 $complete(e)$，Hold 制約を満たすために，信号が到着できる最も早い時間 $hold(e)$，Setup 制約を満たすために，正確な信号にならなければならない最も遅い時間 $setup(e)$ から定められる [6](図 5).

既存手法により限界最小クロック周期を必ず達成する [6] が，挿入する遅延量は，必ずしも少なくない．これは，Step2 のクロックスケジューリングでは Hold 制約を無視しているので，Hold 制約を考慮して適切なクロックスケジュールを与えれば Hold 制約を違反しないレジスタ対も Hold 制約を違反することがあるからである．既存手法 [6] では，Step2 で定められたクロックスケジュールにおいて Hold 制約を違反するレジスタ対が多く，そのクロックスケジュールを変更しないため，違反する Hold 制約を解消するための遅延挿入が多い．また，遅延を挿入する箇所も回路の構造を無視して決定するので，遅延挿入が多い．

3 提案手法

既存手法 [6] では，クロック周期の最小化を目指していたため，目標のクロック周期は，限界最小クロック周期であり，既存手法 [6] の Step2 では，クロック周期を限界最小クロック周期に設定し，クロックスケジュールを決定した．本稿では，所望のクロック周期が与えられた場合，できるだけ少ない遅延挿入量で所望のクロック周期を実現する論理回路を得る手法を検討する．既存手法 [6] の Step2 のクロック周期を所望のクロック周期に設定することで所望のクロック周期を実現する論理回路を得る問題に適用できるので，既存手法 [6] をベースに，遅延挿入量最小化を目指した初期のクロックスケジュールを決定する手法，および，遅延挿入手法を提案する．提案手法では，既存手法 [6] の Step2 から得られる初期のクロックスケジュールを Hold 制約を満たさないレジスタ対の数が少なくなるように修正し，修正したクロックスケジュールの制約を全て満たすように回路の構造を考慮した遅延の挿入を行う．

3.1 クロックスケジュール修正手法

クロックタイミング $S(a), S(b)$ が与えられた時，制約グラフ上の有向枝 (a, b) に対して，レジスタ対 (a, b) に課される制約の余裕度 $\Delta(a, b)$ を

$$\Delta(a, b) = S(a) + w(a, b) - S(b)$$

と定義する．ここで，$w(a, b)$ は制約グラフ上の有向枝 (a, b) の枝重みである．$\Delta(a, b) \geq 0$ のとき，Setup 制約もしくは Hold 制約を満たし，$\Delta(a, b) < 0$ のとき，Setup 制約もしくは Hold 制約を違反する．

既存手法 [6] の Step2 のクロック周期を所望のクロック周期に設定して得られるクロックスケジュールは，全ての Setup 制約を満たしているが，Hold 制約を満たしているとは限らない．Hold 制約を満たしていないレジスタ対数を最小化しても必ずしも遅延挿入量が最小になるとは限らないが，Hold 制約を満たしていないレジスタ対の数が少ないクロックスケジュールを与えると，遅延挿入量が少なくなると考えられる．本節では，Step2 で与えられたクロックスケジュールから，レジスタ対の余裕度 Δ が負のレジスタ対数が少なくなるようにクロックスケジュールを修正する手法を提案する．

ここで，クロックスケジュール S からレジスタ b のクロックタイミングを β だけ下げたクロックスケジュール S' の余裕度について考える (図 6)．クロックスケジュール変更後のレジスタ a, b, c のクロックタイミングを $S'(a) = S(a), S'(b) = S(b) - \beta, S'(c) = S(c)$ とする．クロックスケジュール変更後のレジスタ対 (a, b)，(b, c) の余裕度 $\Delta'(a, b)$，$\Delta'(b, c)$ は，

$$\begin{aligned} \Delta'(a, b) &= S'(a) + w(a, b) - S'(b) \\ &= S(a) + w(a, b) - S(b) + \beta \\ &= \Delta(a, b) + \beta \end{aligned}$$

$$\begin{aligned} \Delta'(b, c) &= S'(b) + w(b, c) - S'(c) \\ &= S(b) - \beta + w(b, c) - S(c) \\ &= \Delta(b, c) - \beta \end{aligned}$$

である．つまり，b のクロックタイミングを β 小さくすると，b から出る有向枝の余裕度も β 小さくなり，b に入る有向枝の余裕度は β 大きくなる．

次に，レジスタ対 (a, b) が制約を違反している $(\Delta(a, b) < 0)$ とき，(a, b) の制約違反が解消されるクロックスケジュールの変更について考える．(a, b) の違反量を $-\Delta(a, b) (> 0)$ とし，$\beta = -\Delta(a, b)$ とする．前

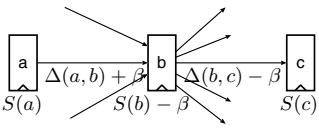


図 6: クロックタイミングの変更.

述より，b のクロックタイミングを β 小さくすると (a,b) の余裕度は β 大きくなり，(a,b) の制約違反が解消される．b に入る有向枝の余裕度は β 大きくなるので，b に入る有向枝に対応するレジスタ対の中で新たに制約違反するレジスタ対は存在しない．一方，b から出る有向枝の余裕度は β 小さくなるので，その有向枝の余裕度が 0 以上 β 未満の場合，その有向枝に対応するレジスタ対は新たに制約を違反する．新たに制約を違反するレジスタ対の制約違反は，その有向枝が入るレジスタのクロックタイミングを違反量だけ下げれば，制約違反が解消される．このようにレジスタのクロックタイミングを入力側の有向枝の違反量だけ下げることでレジスタの入力側から出力側へ違反量を移す作業を，新たに制約違反が起きなくなるまで繰り返すことができると，(a,b) の制約違反を解消でき，(a,b) 以外の違反しているレジスタ対の数も増えることはない．つまり，新たに制約を違反するレジスタ対がなくなるまで，b の出力コーンのクロックタイミングを下げるという操作が a に適用されなければ，(a,b) の制約違反を解消でき，違反しているレジスタ対の数が少なくとも 1 減る．b の出力コーンのクロックタイミングを下げるという操作が a に適用される場合，余裕度が負の枝が (a,b) だけである (a,b) を含む負閉路が存在することになり，クロックタイミングは変更をしても (a,b) の制約違反を解消できないので，クロックタイミングは変更しない．なお，b のクロックタイミングを β 小さくすると，b から出る有向枝の余裕度が負の場合，違反量が大きくなるが，新たに制約を違反することがなく，違反しているレジスタ対の数は多くならない．

上記の操作は，余裕度が負の有向枝の枝重みを十分に大きい正の値，余裕度が 0 以上の有向枝の枝重みを余裕度としたときのレジスタ b からの距離から b に入る違反度 β を引いた値が負のレジスタに対して，その分だけクロックスケジュールを小さくすることに対応する．図 7 に余裕度が負の枝の枝数を極小化する発見的手法を提案する．

3.2 遅延挿入手法

既存手法 [6] では，Step2 で定められたクロックスケジュールを固定し，そのクロックスケジュールの下で制約を違反を解消するために，ある枝に対し遅延余裕量と遅延要求量の小さい方の遅延量を挿入する操作を繰り返すことで，全ての回路で限界最小クロック周期を達成することを明らかにした．しかし既存手法 [6] は，回路グラフで違反レジスタから幅優先探索をし，最初に探索された遅延挿入ができる枝を遅延挿入箇所としており，回路全体の構造を考慮していないため，挿入する遅延量が多い．そこで，本稿では，文献 [8] で提案された，マルチサイクルパスの回路があるクロック周期で正常動作しない時，挿入できるゲートの面積を容量としたグラフの最小カットに含まれる枝にレジスタもしくは遅延を挿入する手法を応用した手法を提案する．

入力： 制約グラフ $H(G,T_L(G))$, クロックスケジュール

出力： 修正後のクロックスケジュール

Step 1： 全ての有向枝に対して余裕度を計算する．

Step 2： 余裕度が負である全ての有向枝に対して，余裕度が負である有向枝の数が減少しなくなるまで，以下の操作を繰り返す．

> 余裕度が負の有向枝の枝重みを十分に大きい正の値，余裕度が 0 以上の有向枝の枝重みを余裕度としたとき，余裕度が負のある有向枝 (a,b) に着目し，全てのレジスタに対して，(変化量)=($\Delta(a,b)$)+(b からの距離) を計算する．変化量が負のレジスタがあれば，そのレジスタのクロックタイミングに変化量を足し，接続するレジスタ対の余裕度を変更する．

Step 3： クロックスケジュールを出力し，終了．

図 7: クロックスケジュール修正手法.

提案手法は，全ての Hold 制約違反が解消されるまで，回路グラフ上の有向枝に遅延余裕量と遅延要求量から容量を設定し，Hold 制約違反をするレジスタ対をソース点，シンク点に設定し，最小カット内の複数の枝に遅延を挿入する手法である (図 8)．一度の操作で全ての Hold 制約を解消できるとは限らないので，この操作は繰り返される．提案手法は遅延挿入箇所を変更しているが，既存手法 [6] と同様に遅延余裕量と遅延要求量の小さい方の遅延量を挿入する操作を繰り返すので，所望のクロック周期を限界最小クロック周期に設定しても所望のクロック周期を実現する論理回路を得られる．

最小カットに含まれる有向枝に遅延を挿入する操作で少ない遅延挿入量で制約違反が解消されるように，遅延余裕量と遅延挿入量を基に回路グラフ上の有向枝 e の容量を設定することを考える．$demand(e) \leq 0$ ならば，遅延を挿入する必要がないので容量 0 にする．e が $slack(e) > 0$ ならば，e を通過するパスの Setup 制約を満たしたままで e に $slack(e)$ の遅延を挿入できる，$demand(e) > 0$ ならば，e を通過するパスの Hold 制約を満たすためには，e を通過するパスの中に合計 $demand(e)$ の遅延を挿入しなければならない．つまり，$0 < demand(e) \leq slack(e)$ ならば，e に $demand(e)$ の遅延を挿入することで e を通過するパス全ての制約が満たされることになり，e に遅延が挿入されることが望ましいので，最小カットに含まれやすくするため容量を 1 にする．$0 < slack(e) < demand(e)$ ならば，e に $slack(e)$ の遅延を挿入できるが，Hold 制約を満たすためには他の場所にさらに遅延を挿入しなければならない．$0 < demand(e) \leq slack(e)$ の有向枝よりも最小カットに含まれて欲しくないが，遅延が挿入できるので，$0 < slack(e) < demand(e)$ ならば，容量を 2 にする．$demand(e) > 0, slack(e) \leq 0$ ならば，最小カット

入力： 回路グラフ G, クロックスケジュール，Hold 制約違反をするレジスタ対集合

出力： 遅延挿入後の回路グラフ

Step 1： 全ての有向枝に対して遅延余裕量，遅延要求量を計算する．遅延要求量が正の枝がない場合は，回路グラフを出力し，終了．遅延余裕量，遅延要求量から容量を設定する．

Step 2： ソース点，ソース付近の容量，シンク点を設定する．

Step 2： 最小カットを求める．最小カット上の枝で，上流に最小カット上の全ての枝に対して，遅延余裕量と遅延要求量の小さい量の遅延値を挿入し，Step1 へ．

図 8: 遅延挿入手法.

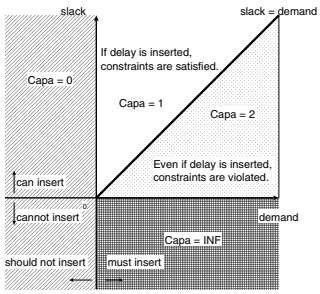


図 9: 遅延余裕量と遅延挿入量と容量の関係.

に含まれても遅延が挿入できないので，最小カットに含まれないように容量を無限にする (図 9).

容量が 2 の有向枝が最小カットに含まれる場合，最小カットに含まれる全ての有向枝に遅延を挿入しても，制約を満たさないレジスタ対が残ってしまう．この制約を満たすようにするため，回路グラフ上の各有向枝に対して遅延余裕量と遅延要求量を再計算し，遅延挿入する操作を繰り返す.

また，最小カットを求めた場合，1 つのパス上に最小カット上の有向枝が 2 つ以上含まれる場合がある (図 10)．遅延余裕量と遅延要求量は，その有向枝の入力方向と出力方向の遅延値により定まるので，1 つのパス上に 2 つ以上の有向枝に遅延を挿入すると，過剰に遅延を挿入したり，Setup 制約を違反したりすることがある．そこで提案手法では，1 つのパス上に 2 つ以上の有向枝

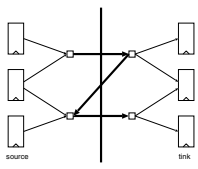
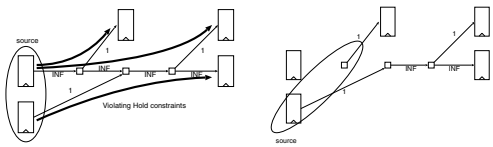


図 10: 1 つのパス上の枝が最小カットに 2 つ以上含まれる例．有向枝の容量は全て 1.



(a) 修正前. (b) 修正後.

図 11: 全ての有向枝 e が $demand(e) > 0$ かつ $slack(e) = 0$ となるパス.

表 1: 回路データと最適解．ただし，s6669 はメモリオーバーになる直前の解と計算時間を示す.

	Size		Clock Period			MILP	
File	$\|V_g\|$	$\|V_r\|$	T_C	T_S	T_L	D_{add}	Time[s]
s298	270	15	9	6	5.334	3	0.02
s344	360	16	20	17	14.000	3	0.03
s349	362	16	20	17	14.000	3	0.02
s444	408	22	11	7	6.584	13	0.02
s526	432	22	9	6	5.500	2	0.05
s635	639	33	127	124	66.000	422	0.03
s991	1142	20	59	55	54.000	1	0.05
s1269	1231	38	35	30	18.667	120	1.04
s1423	1480	75	59	54	53.000	1	0.08
s1512	1704	58	30	24	22.500	4	0.11
s3271	3403	117	28	19	14.715	48	0.75
s3384	3780	184	60	51	26.500	256	1.44
s4863	4942	105	58	53	30.000	552	0.63
s6669	6722	240	93	81	25.167	2487	2729.10
prolog	3511	137	26	14	12.500	9	0.37
s15850	20753	598	82	57	42.000	82	2.60
s15850.1	20690	535	82	71	63.000	8	1.50
s35932	35622	1729	29	28	27.000	1	3.53

$|V_g|$ 回路グラフの点数
$|V_r|$ 制約グラフの点数
T_C 完全同期方式の最小クロック周期
T_S 一般同期方式の最小クロック周期
T_L 限界最小クロック周期
D_{add} 限界最小クロック周期を達成する時の遅延挿入量の最小値
Time[s] MILP の定式化の計算時間 + MILP を解く計算時間

が最小カットに含まれた場合は，そのパス上のソース点に近い有向枝に遅延を挿入し，そのパス上のそれ以外の点には遅延を挿入しない.

あるパスが Hold 制約が違反する場合，そのパス上に slack が正の有向枝が存在する [6] が，Hold 制約が違反する複数のパスにより，パス上の全ての有向枝 e が $demand(e) > 0$ かつ $slack(e) = 0$ となるパスが存在することがある (図 11(a))．このようなパスが存在する場合，このパス上の全ての有向枝の容量が無限になるので，最小カットに容量が無限の枝が含まれ，遅延が挿入できない．また，この場合は，この有向枝以外の最小カットに含まれる全ての有向枝に遅延を挿入し，その後に遅延挿入する操作を繰り返しても，必ず最小カットに容量が無限の枝が含まれ，制約違反が解消されない．そこで提案手法では，最小カットに容量が無限の枝が含まれないようにするため，ソース点から容量が無限の有向枝が出ている場合，その枝を除去し，除去した枝の接続点が入次数が 0 の場合，その点をソース点に加える (図 11(b))．このようにグラフを修正することで，ソース点から出る有向枝の容量の総和は有限となり，無限のフローが流れることがないので，最小カットに容量が無限の枝が含まれることがない.

4 計算機実験

提案手法の有効性を確かめるため，既存手法 [6]，文献 [11, 7] を基に定式化した数理計画法を用いる手法，提案手法を ISCAS89 ベンチマーク回路に対して適用する実験を行った．全ての論理ゲートの遅延，挿入するゲート

表 2: 実験結果．MILP による最適解と計算時間で正規化して示す．

	初期+幅優先 (既存 [6])		初期+最小カット		修正+幅優先		修正+最小カット (提案)	
File	D_{add} [%]	Time [%]	D_{add} [%]	Time [%]	D_{add} [%]	Time [%]	D_{add} [%]	Time [%]
s298	433.33	0.00	266.67	0.00	100.00	0.00	100.00	0.00
s344	2300.00	0.00	2300.00	33.33	100.00	0.00	100.00	33.33
s349	2300.00	0.00	2300.00	50.00	100.00	0.00	100.00	0.00
s444	153.85	50.00	130.77	0.00	107.69	50.00	100.00	50.00
s526	650.00	20.00	400.00	0.00	150.00	0.00	150.00	0.00
s635	309.00	33.33	309.00	0.00	100.00	0.00	100.00	33.33
s991	629300.00	560.00	164900.00	40.00	100.00	0.00	100.00	0.00
s1269	512.50	7.69	421.67	3.85	260.83	5.77	164.17	1.92
s1423	309000.00	425.00	282500.00	87.50	100.00	37.50	100.00	37.50
s1512	34875.00	336.36	25600.00	72.73	250.00	9.09	100.00	9.09
s3271	3475.00	189.33	1706.25	12.00	141.67	32.00	112.50	5.33
s3384	901.95	62.50	778.91	5.56	169.14	27.08	150.78	4.17
s4863	1128.44	1298.41	742.21	320.63	388.59	517.46	227.72	109.52
s6669	461.00	0.76	288.22	0.13	341.46	0.52	225.77	0.10
prolog	5711.11	254.05	1511.11	13.51	177.78	24.32	100.00	5.41
s15850	12798.78	2991.54	6007.32	123.85	390.24	301.54	146.34	61.92
s15850.1	272312.50	6655.33	176875.00	417.33	100.00	42.00	100.00	41.33
s35932	429000.00	6700.00	429000.00	173.65	100.00	27.76	100.00	26.63
AVE.	94756.80	1088.02	60890.95	75.23	176.52	59.72	126.52	23.31

の遅延を 1 とし，レジスタの遅延，配線遅延を 0 とした．この場合，数理計画法は混合整数計画問題となる．混合整数計画問題 MILP は，ILOG 社の CPLEX10.0.0[1] を用いて解いた．その他の手法は 3.40GHz/1GB Intel Pentium-4 CPU，1GB RAM の PC に gcc3.5.5 の C++ で実装した．所望のクロック周期は限界最小クロック周期とする．

ISCAS89 ベンチマーク回路の 48 回路において，一般同期方式の最小クロック周期と限界最小クロック周期が一致する回路が 30 回路あった．この 30 回路にどの遅延挿入手法を適用しても，遅延を挿入することはない．そこで，一般同期方式の最小クロック周期よりも限界最小クロック周期の小さい 18 回路に対して提案手法を適用した．回路のデータと混合整数計画問題 MILP による最適解を表 1 に示す．ただし，s6669 はメモリオーバーにより，最適解が求まらなかった．そこで，表にはメモリオーバーになる直前の時間と解を示す．

クロックスケジューリング手法として既存手法 [6] のクロックスケジュールを用いる場合 (初期) と提案クロックスケジュール修正手法 (修正) の 2 手法を適用し，遅延挿入手法として遅延挿入箇所を幅優先による探索する既存手法 [6](幅優先) と最大フロー・最小カット法により探索する提案手法 (最小カット) の 2 手法を適用した．4 手法の遅延挿入量と計算時間を表 2 に示す．表 2 では，MILP による最適解と計算時間で正規化して示す．

既存手法 (初期+幅優先) は遅延挿入量が最適解より約 948 倍となり，遅延挿入量が多いため計算時間も長い．遅延挿入箇所の選定のみを提案手法に変更しても (初期+最小カット)，遅延挿入量が最適解より約 609 倍である．他方，クロックスケジュールだけを変更しただけで (修正+幅優先)，遅延挿入量が最適解の 76.5%増と大きく減少し，提案手法 (修正+最小カット) では，遅延挿入量が最適解の 26.5%増で半数以上の回路で最適解を達成する．また，遅延挿入回数が抑えられるので，計算時間も大幅に減少する．

5 まとめと今後の課題

そこで本稿では，各ゲートの最大遅延値と最小遅延値が等しい場合，既存手法 [6] を基にクロック入力タイミングを修正する発見的手法と，遅延挿入箇所の決定する発見的手法を提案した．計算機実験において，既存手法より遅延挿入量を大幅に改善し，多くの回路で挿入する遅延量が最適解と等しくなることを示した．

今回は，回路モデルを簡単化したため，1 回路を除いて最適解が求めることができたが，実際の回路では，厳密解を求めるのは困難であると考えられる．今後の課題として，より実用的な回路モデルに対する遅延挿入手法を提案し，提案手法の有効性を確かめることである．

謝辞 本研究は一部科学研究費補助金 (特別研究員奨励費) (課題番号:19·6015) によるものである．

参考文献

[1] ILOG CPLEX10.0, http://www.ilog.co.jp/.

[2] R. B. Deoker and S. S. Sapatneker. A Graph-Theoretic Approach to Clock Skew Optimization. In *ISCAS*, pp. 407–410, 1994.

[3] J. P. Fishburn. Clock skew optimization. *IEEE Tranc. on Computers*, Vol. 39, No. 7, pp. 945–951, 1990.

[4] K. Inoue, W. Takahashi, A. Takahashi, and Y. Kajitani. Schedule-Clock-Tree Routing for Semi-Synchronous Circuits. *IEICE Trans. Fundamentals*, Vol. E82-A, No. 11, pp. 2431–2439, 1999.

[5] 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, 2002.

[6] 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, 2005.

[7] C. Lin and H. Zhou. Clock Skew Scheduling with Delay Padding for Prescribed Skew Domains. In *ASP-DAC*, pp. 541–546, 2007.

[8] B. A. Rosdi and A. Tkahashi. Delay Balancing by Min-Cut Algorithm for Reducing the Area of Pipelined Circuit. 第 20 回回路とシステム軽井沢ワークショップ, pp. 643–648, 2007.

[9] A. Takahashi. Practical Fast Clock-Schedule Design Algorithms. *IEICE Trans. Fundamentals*, Vol. E89-A, No. 4, pp. 1005–1011, 2006.

[10] A. Takahashi and Y. Kajitani. Performance and Reliability Driven Clock Scheduling of Sequential Logic Circuits. In *ASP-DAC'97*, pp. 37–43, 1997.

[11] B. Taskin and I. S. Kourtev. Delay Insertion Method in Clock Scheduling. *IEEE trans. CAD*, Vol. 25, No. 4, pp. 651–663, 2006.

[12] 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, 1999.