修 士 論 文

高性能LSIのためのポストシリコン・スキュー調整の方式と最適化に関する研究

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

LI Jian

2011年9月

修 士 論 文

高性能LSIのためのポストシリコン・スキュー調整の方式と最適化に関する研究

指導教員　金子　峰雄　教授

審査委員主査　金子　峰雄　教授
審査委員　田中 清史　准教授
審査委員　井口　寧　准教授

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

0910211 LI Jian

提出年月: 2011 年 8 月

概 要

LSI 製造技術の発展、特に半導体微細化の進展に伴い、製造ばらつきとそれによる歩留まり低下、性能劣化が LSI 設計における最も重要な問題の一つになっている。デジタル集積回路において、製造ばらつきは動作タイミングのずれとなって現れ、回路の正しい動作を阻害することになる。この製造ばらつきによるタイミング誤りの問題を解決するための手法はプリ・シリコン（製造前）の設計手法とポスト・シリコン（製造後）調整手法の 2 つに分けられる。本研究では、予め回路に組み込まれた Programmable Delay Element(PDE) を使って LSI 製造後にタイミングスキューを調整することにより、製造ばらつきに対して高い性能歩留まりを確保する方式を対象に、製造後のチップ個別に、PDE 調整量を決定する手法を開発した。これは従来の遅延量計測と数値計算による PDE 調整量決定と異なり、PDE 制御とタイミングテストを繰り返すことにより PDE 調整を行うものである。シミュレーション実験によりタイミング誤りが存在する回路の歩留まりを改善する有効性を確認した。

目 次

第 1 章　はじめに **1**

第 2 章　タイミング制約と PDE の導入 **3**
2.1　セットアップとホールド制約条件 3
2.2　PDE の導入 3
2.2.1　PDE 回路 3
2.2.2　PDE モデルと効果 5
2.3　既存研究 7

第 3 章　提案手法 **10**
3.1　モデル 10
3.2　発想 11
3.3　相対補正制約有向グラフ 13
3.4　アルゴリズム 14
3.4.1　アルゴリズムの全体像 14
3.4.2　制約グラフ G の初期化 15
3.4.3　タイミングテスト 16
3.4.4　辺の着色と w の更新 16
3.4.5　第一回の離散制御信号値の設定 18
3.4.6　第二回以降の離散制御信号値の設定 23
3.4.7　アルゴリズムの有効範囲 24

第 4 章　実験結果と分析 **28**
4.1　調整性能に関する実験 28
4.2　最小タイミングマージンの上界に関する実験 30
4.3　PDE の遅延曲線に関する実験 31

第 5 章　まとめと今後の課題 **36**

第1章　はじめに

LSI 製造技術の発展、特に半導体微細化の進展に伴い、製造ばらつきとそれによる歩留まり低下、性能劣化が LSI 設計における最も重要な問題の一つになっている。デジタル集積回路において、製造ばらつきは動作タイミングのずれとなって現れ、回路の正しい動作を阻害することになる。現在において製造されたチップの欠陥の中でタイミングエラーが30%を占めると言われる [1]。一方で、こうした製造ばらつきによるタイミングエラーを回避するための設計マージンの確保は、集積回路の高性能化を難しくている。

この製造ばらつきによるタイミングエラーの問題を解決するための手法は大きく二つに分けられる。一つはプリ・シリコン（製造前）の設計手法であり、他の一つがポスト・シリコン（製造後）調整手法である。

前者の代表的な手法は統計的静的遅延解析（SSTA：Statistical Static Timing Analysis）とそれに基づく回路の最適化である [2]。これは、トランジスタ等のデバイスの特性パラメータの確率分布（PDF:probability distribution function)、あるいはゲート回路や配線の信号伝搬遅延の確率分布などからシステム特性の統計的な確率分布を計算し、特性と歩留まりをトレードオフする手法であるが、デバイスやゲート回路特性の正確な確率を得るのが難しく、また製造後のチップの性能を十分に引き出せていないなどの問題もある。

一方、後者のポスト・シリコン調整は製造されたチップ毎に、観察できる実際の情報を用いて、回路を物理的に調整することから、製造後のチップ個別の性能を十分に引き出せる有効的で、実用的な方法である。こうしたポスト・シリコン調整の代表的な手法として、クロックスキュー（クロック分配遅延）調整がある [4][5][6][7][8]。

LSI の製造後のクロック調整手法では、予め回路にクロック分配遅延量を調整するための PDE（Programmable Delay Element）を組み込んでおき、チップ製造後に、測定機器などによって各チップから必要な情報、例えば、レジスタ間の最大遅延と最小遅延などを計測・情報収集して、ホストコンピュータ上で PDE の設定値を計算し、回路中の PDE にその値をセットする。この手法の実用化には幾つかの重要な問題がある。その一つはスキュー調整に必要十分な情報を選択、入手する方法そのものである。基本的には、全ての回路状態 (MUX の入力選択の全通り) における全てのレジスタ間の信号伝搬に関する最大遅延と最小遅延とが計測できれば、スキュー調整量の計算に支障はないが、この計測には膨大なコストが掛かり、実用的とは言えない。他の一つは PDE のセット可能な遅延値が離散的な値をとり、且つそれら自身が製造ばらつきの対象となることである。原則的には、回路中の全ての PDE について、それがセット可能な全ての離散的な遅延量を最大遅延と最小遅延の観点から計測できる必要があり、先の遅延情報収集とも相俟って、そのコ

ストは膨大なものとなる。

本稿では、従来の遅延量計測と数値計算による PDE 調整量決定と異なり、タイミングテストと PDE の制御信号値調整を繰り返すことにより PDE 調整量を決定する手法を提案する。始めに FF(PDE) を頂点とし、FF 間のタイミング制約を辺とする有向重み付きグラフ (相対補正制約有向グラフ) を定義する。辺の重みは二つ FF 間のスキューを調整する最小必要な調整量の差とする。辺の色 (赤と緑) は相対的に一方の FF のクロック到着を他方の FF に対して遅らせる必要があるかどうかを表す。具体的な PDE 調整ではまず相対補正制約グラフ G を構築する。毎回タイミングテストした後テストから得られタイミングエラーの種類と対応する FF の情報を用いて各辺を赤あるいは緑に着色し、辺重み w を更新する。この際に制約グラフ G が辺重みについての正サイクルを持つとき、如何なる PDE 離散制御信号値の設定によっても、タイミング違反をすべて解消することはできないので、調整不可能として終了する。正サイクルを持ってなければ、グラフの最長路問題として各辺の最小必要な差 w(辺重み) を満たす頂点離散制御信号値を求める。各頂点の離散制御信号値を回路の PDE に適用してもう一度タイミングテストを実施する。全てのタイミング誤りを解決するまでに繰り返す。

本稿の構成は以下の通りである。第 2 章では、タイミング制約と PDE 回路、モデル及び効果を説明する、第 3 章では、PDE 制御とタイミングテストを繰り返すことにより PDE 調整を行う提案アルゴリズムを説明する。第 4 章では、実験方法及び実験内容を説明し、実験結果を分析する。第 5 章はまとめと今後の課題である。

第2章　タイミング制約とPDEの導入

2.1　セットアップとホールド制約条件

デジタル集積回路は入出力関係がブール関数で与えられる組み合わせ論理回路と書き込み制御信号を受けて値を読み込み・保持するフリップフロップ (FF) とから成っている。正しく値を FF に書き込むために、FF と FF の間にセットアップ条件とホールド条件を守らなければならない。

セットアップ条件は、値が書き込む先の FF に到着する前に、書き込み制御信号が到着してはいけないことである。ホールド条件は、FF への書き込み制御信号が到着する前に、値が失われてはいけないことである。

セットアップ時間はクロック信号立ち上がり時刻よりも前もって入力信号が確定されなければならない時間である。ホールド時間はクロック信号立ち上がり後ある一定時間の間、入力値が変化しないことが要求される時間である。

図 2.1 にセットアップとホールド制約条件を示す。t_a と t_b それぞれは FF_a と FF_b の製造後の書き込み制御信号の到着時刻である。Δ_{ab} は FF_a と FF_b の間にあるパスの信号伝搬遅延である。CP はクロック周期である。T_S は FF のセットアップ時間で、T_H は FF のホールド時間である。FF のセットアップ時間とホールド時間を含めて、セットアップ条件を守るために FF_a からの値が $t_b + CP - T_S$ 前に書き込む先の FF_b に到着しなくてはいけない。ホールド条件を守るために $t_b + T_H$ 前に前の FF_a からの値が失われてはいけない。

式 (2.1)(2.2) で示すセットアップ制約条件：

$$t_a + \Delta_{ab} \leq t_b + CP - T_S \tag{2.1}$$

ホールド制約条件：

$$t_a + \Delta_{ab} \geq t_b + T_H \tag{2.2}$$

2.2　PDEの導入

2.2.1　PDE回路

PDE 回路の設計手法は幾つかある。一つは図 2.2 に示す「tapped delay line block」[3] であり、最小単位遅延の 0 倍、1 倍、2 倍、3 倍の遅延量を発生できる。上の部分は遅延

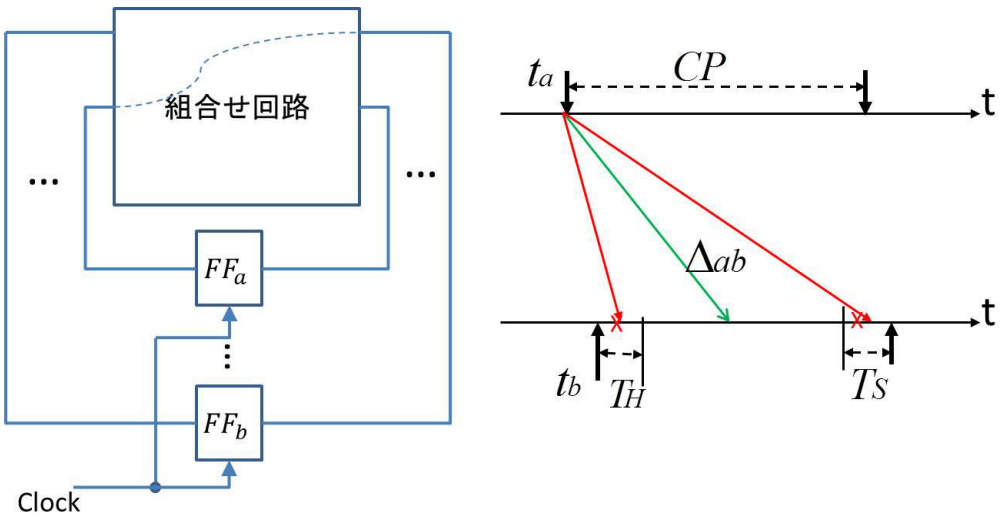

図 2.1: タイミング制約

バッファチェインである。「delay」は二つのインバーターからなり、最小遅延量を発生する。クロック出力は下のマルチプレクサに遅延バッファチェイン上の異なる節点と接続されている。コントロール信号によりこの節点を選択するので、最小単位遅延の整数倍遅延量を発生できる。この PDE 回路の欠点は全部の遅延バッファチェインが常時動作しているので、消費電力が大きいことである。更に、コントロール信号の数は遅延値の数と同じにしなければならないので、回路の規模が大きくなってしまう。文献 [7] では図 2.3 のよ

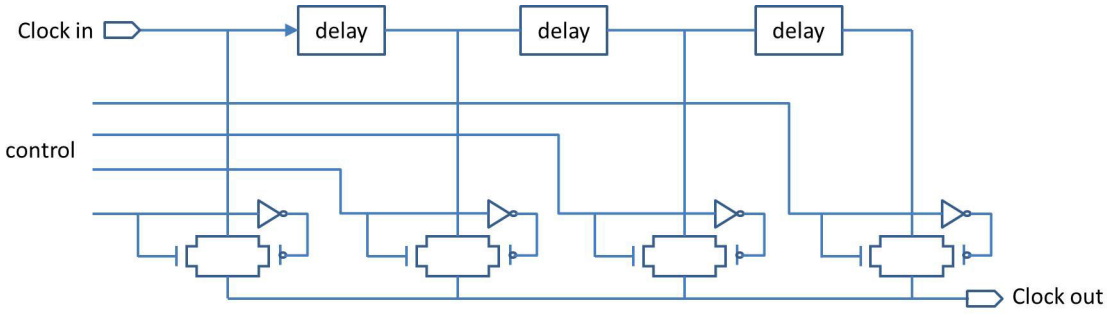

図 2.2: tapped delay line block

うな PDE を導入している。クロック入力とクロック出力の間に二つインバーターが設置されている。DAC(Digital-Analog Converter) は入力信号ベクトルにより異なる電圧 V_{out} を発生する。この電圧は上の二つインバーターの充電電流と放電電流をコントロールしているので、異なるコントロール電圧に対して異なる遅延量が生じる。しかし、この方法で発生された遅延量はコントロール電圧値に対して非線形である。図 2.4 に示す PDE 回路は入力信号ベクトルにより伝送ゲートとそれらの容量負荷の数を調整しながら遅延量を

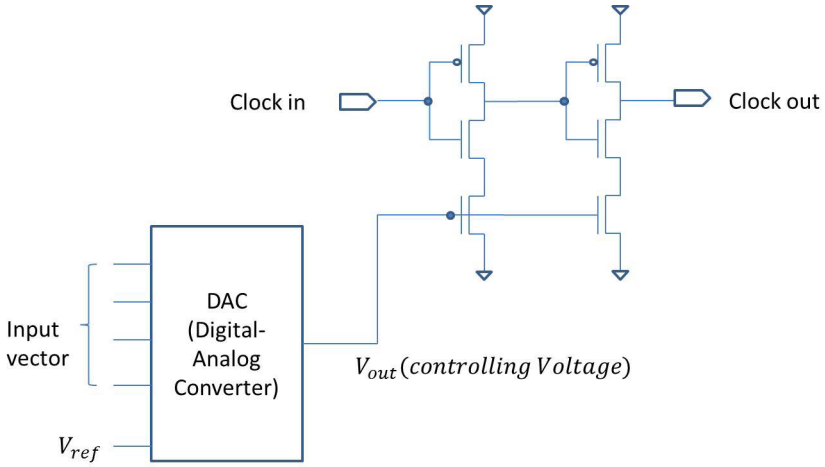

図 2.3: PDE 回路の一つ例-電圧コントロール

コントロールする。遅延値の曲線が制御値に対してほぼ線形であることはこの回路の一つ大きな利点である。また、コンデンサ容量を調整するだけで遅延値の範囲を調整できることも利点の一つである。

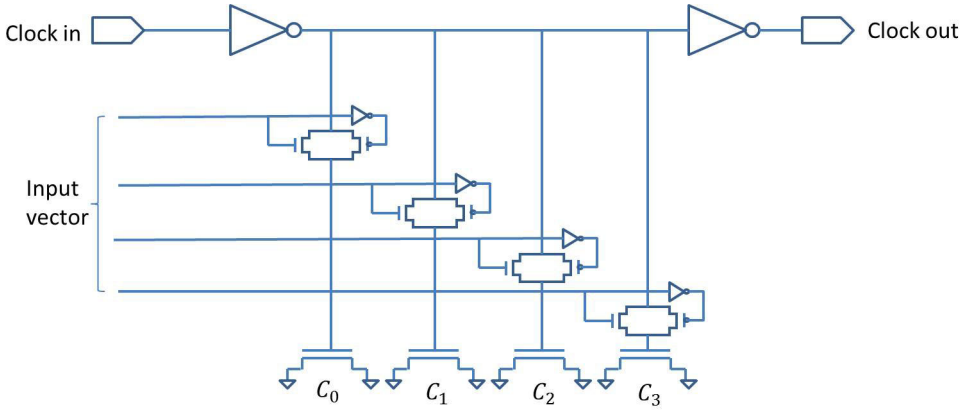

図 2.4: PDE 回路の一つ例-負荷容量コントロール

2.2.2 PDE モデルと効果

PDE の物理特性は設計手法により異なるが [9]、理論上図 2.5 に示されている PDE のモデルを使う。PDE は主に二つ部分がある。制御信号値を保持するレジスタ部分と遅延を発生する部分である。一つのレジスタ値に対して一つ遅延が対応している。制御値は離散的であるから、遅延値も離散的となる。更に PDE 自体が集積回路上に形成されることから、ばらつきの対象となっている。これらは実用化に対する無視できない性質である。

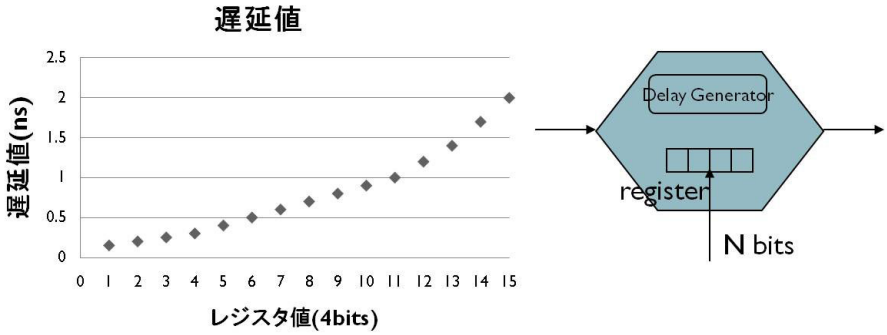

図 2.5: PDE の特性

図 2.6 のように予め回路にクロック分配遅延値を調整するための PDE(Programmable Delay Element) を組み込んでくとき、正しい動作のためのタイミング条件 (セットアップ、ホールド条件) は式 (2.3)(2.4) のようになる。ここで d_a は PDE_a により FF_a でのクロック到着時刻は遅延値 d_a だけ遅れる。d_b は PDE_b により FF_b でのクロック到着時刻は遅延値 d_b だけ遅れる。

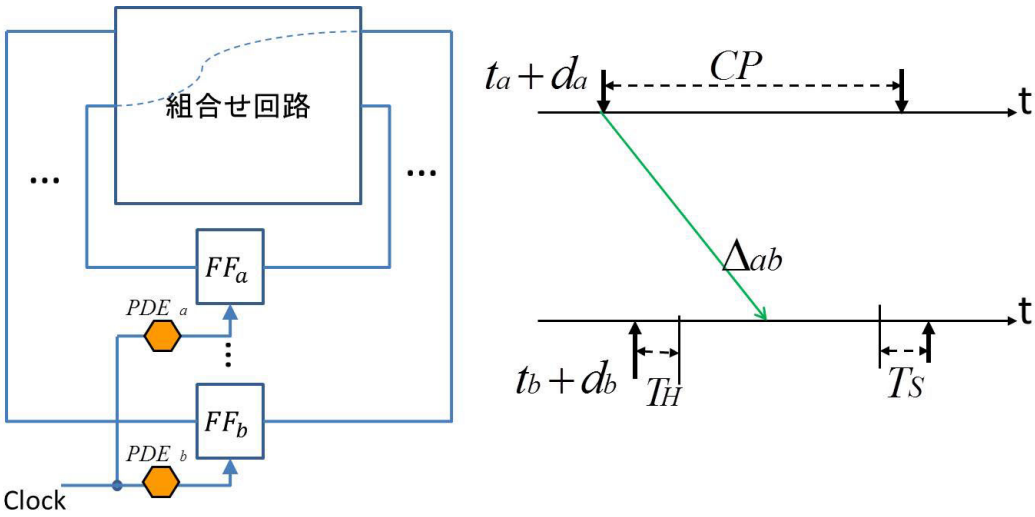

図 2.6: PDE 導入後のタイミング制約

$$t_a + d_a + \Delta_{ab} \leq t_b + d_b + CP - T_S, \tag{2.3}$$

$$t_a + d_a + \Delta_{ab} \geq t_b + d_b + T_H \tag{2.4}$$

図 2.7 に PDE の効果を表す一つの例を示す。左側に PDE が挿入されない時、$t_a + \Delta_{ab} = 2.8ns$, $t_b + CP - T_S = 2.4ns$ であることから, 明らかにセットアップ制約条件を違反して

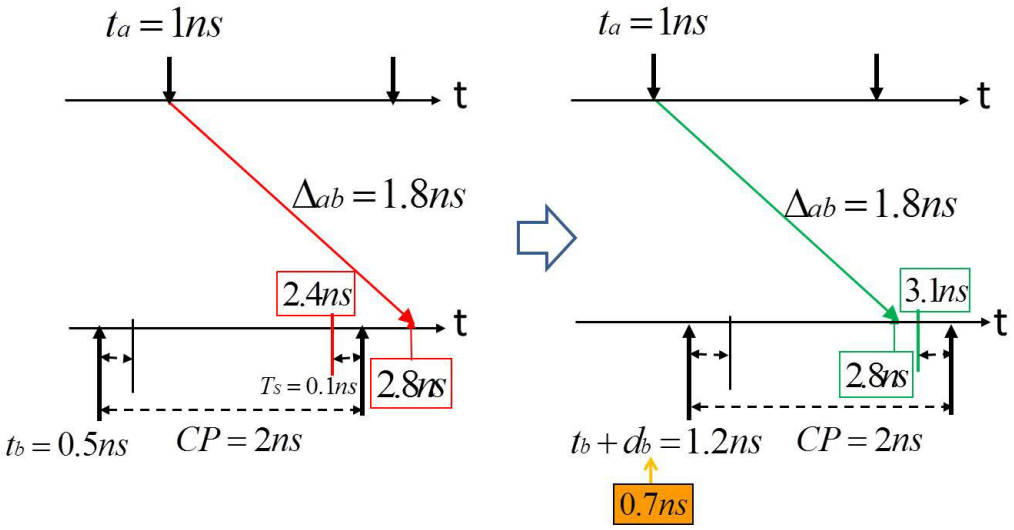

図 2.7: PDE の効果を示す一つ例

いる。右側の図が示すように FF_b の前に PDE が挿入され、PDE の遅延値を 0.7ns と設定すれば、$t_b + d_b + CP - T_S = 3.1ns$ となり、左側のセットアップ制約違反が解消された。

2.3 既存研究

製造ばらつきによるタイミングエラーの問題を解決するため、LSI 製造後に回路に組み込まれた Programmable Delay Element(PDE) を使ってタイミングスキューを調整することにより、製造ばらつきに対して高い性能歩留まりを確保する方式が考えられている。この際に、一つ一つのチップに対してどのように PDE の調整を行うかが問題となる。

一つ手法は遅延量の計測に基づくものである [4][5]。

[4] はチップ製造後に測定機器などによって FF 間の信号伝搬遅延 Δ_{ij}、FF のクロック信号到着時刻 t_i, t_j を得られると仮定する。セットアップ制約条件 $t_i + d_i + \Delta_{ij} \leq t_j + d_j + CP - T_S$ とホールド制約条件 $t_i + d_j + \Delta_{ij} \geq t_j + d_j + T_H$ の中に未知数は PDE の遅延値 d_i と d_j のみであるから、LP(linear programming) 定式化して各 PDE の遅延値を決める。

[5] は計測のコストを削減するために製造ばらつきは正規分布に従うとの仮定の下で一部の FF におけるクロック到着時刻を測定し、その値に基づき残りの FF のクロック到着時刻を推定するスキュー調整手法を提案している。クロック信号に対するクロック分配モデル H-tree の各枝の遅延は正規分布に従い、各枝間の遅延の分布はそれぞれ独立であると仮定する。その結果、クロックソースから各 FF までのクロック信号は平均がクロックソースから各 FF まで通過する枝の平均の和、分散が通過する分散の和であるような正規分布となると仮定して、図 2.8 のように t_i の測定結果から t_j の範囲を推定する。この範囲を使って LP(linear programming) 定式化して各 PDE の遅延値を決める。

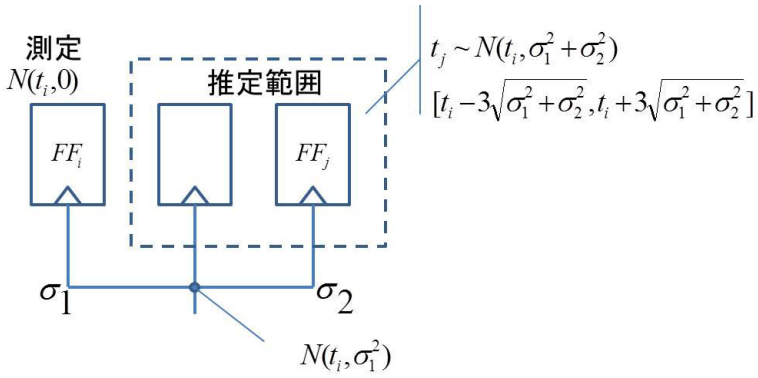

図 2.8: 論文 [5] の手法

[6] はクロックタイミングの測定を実行せずにスキュー調整を実現する手法 (図 2.9) を提案する。まず、各 PDEcluseter ペアの間に多くても一つ critical パスを選ぶ。そして critical パスを対象に異なるクロック周期$CP_0, CP_1, \ldots, CP_{max}$、但し$CP_0 < CP_1 < \ldots < CP_{max}$、で遅延テストする。テストの結果は次の三種類に分けられる。①CP_0 で成功した、②CP_k で失敗したがCP_k+1で成功した、③CP_{max} で失敗した。それぞれの場合に対して$t_i-t_j+\Delta_{ij}$の値を見積もる。PDE の遅延値を求める時、離散的なスキュー値を仮定し、推定した $t_i - t_j + \Delta_{ij}$ の範囲を入れて ILP（整数線形計画法）にて各 PDE の遅延値を決める。

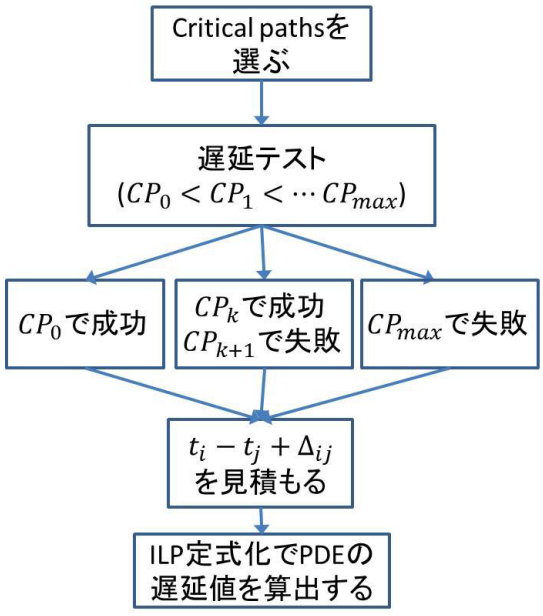

図 2.9: 論文 [6] の手法

これらの手法の問題は計測に膨大なコストが必要となること。連続的スキュー値を仮定

する手法ではPDEの離散的遅延量対応できないこと。離散的なスキュー値の仮定ではスキュー値の計算時間が膨大であり、且つPDEのばらつきに対応できないことである。

もう一つ観点はチップの動作テストにおいて、PDE値セットと動作確認を繰り返すことによって目的のPDE値をセットすることである。

[7]にはチップの動作テストにおいて、成功したテストベクタの数を適応度とし、PDEの遅延値を遺伝子とした遺伝的アルゴリズムとして定式化されている手法(図2.10)が提案されている。しかし、遺伝的アルゴリズムのコストが膨大であり、且つ解の保証がない。

[8]は元々回路性能改善を目的として異なるいくつかのクロック周期で繰り返すものであるが、その中の一つの固定されたクロック周期に対するPDE調整の流れを図2.10の右側に示す。回路を遅延テストした後CPT(Critical Path Tracing)を用いてクロック信号到着時刻が遅くなるべきFFの集合と早くなるべきFFの集合を見つける。PDEの遅延値はこの二つFFの集合により調整する。すべてのタイミングエラーが無くなるまでこの操作を繰り返す。しかし、この[8]ではタイミングエラーとしてセットアップ制約条件違反のみを想定しており、ホールド制約を考慮していない。

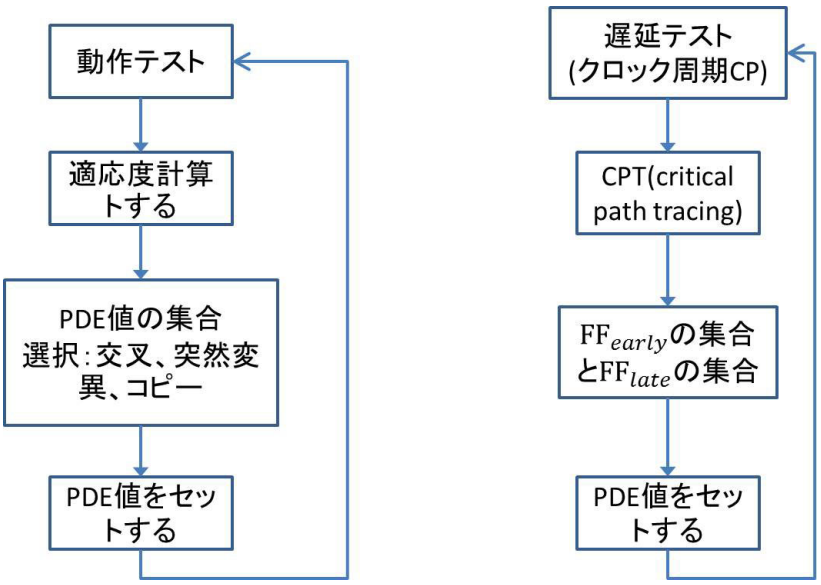

図 2.10: 論文[7]と[8]の手法

第3章　提案手法

3.1　モデル

順序回路における、一般的には、クロック分配モデルはよく H-tree と仮定するが、本稿では、提案手法の特性により、任意のクロック分配モデルに対応できる。図 3.1 のように回路に三種類のばらつきが発生する。クロック信号の到着時刻にばらつきが生じ、FF 間の伝搬遅延にばらつきが生じ、PDE 自身の遅延値にばらつきが生じると仮定する。各 FF のクロック信号線に一つの PDE を挿入することで、FF 毎にクロック信号到着時刻を調整する。

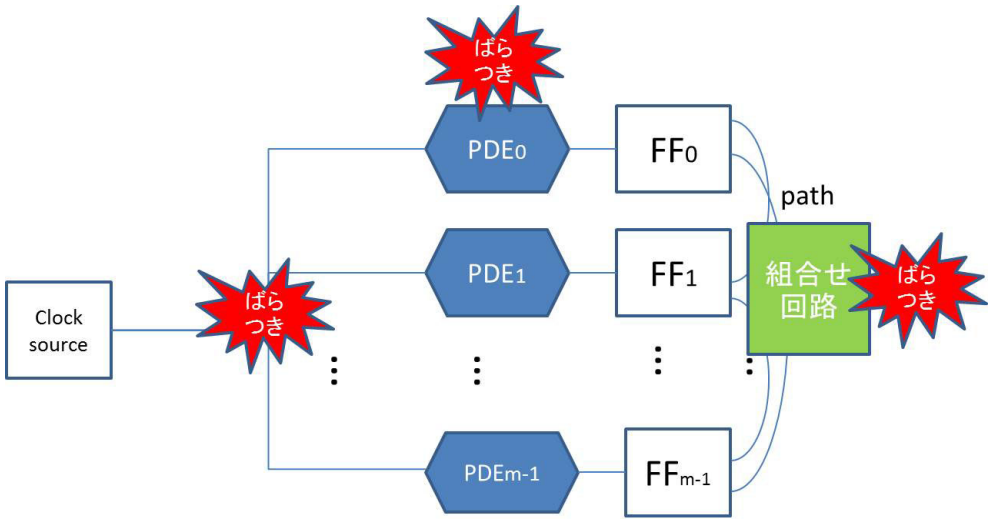

図 3.1: モデル

FF の集合は式

$$FF = \{FF_0, FF_1, \ldots, FF_{m-1}\} \tag{3.1}$$

で定義される。

PDE の集合は式

$$PDE = \{PDE_0, PDE_1, \ldots, PDE_{m-1}\} \tag{3.2}$$

で定義される
PDE_i のレジスタの値は

$$R_i = \{0, 1, \ldots, 2^{Nbit} - 1\} \tag{3.3}$$

で定義される。ここで、$Nbit$ はレジスタのビット数である。
PDE_i の遅延量 d_i は

$$d_i = f_i(R_i) \tag{3.4}$$

$$\tag{3.5}$$

にて与えられる i 番目の FF(FF_i) に対して挿入された PDE_i は離散制御信号値 R_i に応じて、遅延 $f_i(R_i)$ を発生する。ここで、$f_i(R_i)$ そのものは製造ばらつきの対象となるが、$f_i(a) < f_i(b), a < b$、を仮定する。
FF 間のパスの集合を

$$PATH = \{P_0, P_1, \ldots, P_{n-1}\} \tag{3.6}$$

とする。
各パス P_i の両端の FF は

$$ffpair(P_i) = (FF_a, FF_b) \tag{3.7}$$

で示される。式 (3.7) はパス P_i が FF_a を始点とし FF_b を終点とするものであることを示している。
各パス P_i のタイミングテスト結果は

$$testResult(P_i) \in \{setupfail, holdfail, bothfail, pass\} \tag{3.8}$$

で示される。ここでタイミングテストの結果により、$setupfail$ はセットアップ制約を違反するが、ホールド制約を満たすことを示す。$holdfail$ はホールド制約を違反するが、セットアップ制約を満たすことを示す。$bothfail$ はセットアップ制約とホールド制約を両方共に違反することを示す。$pass$ はセットアップ制約とホールド制約を両方共に満たすことを示す。

3.2 発想

まず、タイミング制約式を考えてみる。式 (2.3) と (2.4) から以下の式に導ける。

$$T_H - \Delta_{ab} \leq (t_a - t_b) + (d_a - d_b) \leq CP - T_S - \Delta_{ab} \tag{3.9}$$

デジタル回路におけるスキュー調整の目的はすべての FF 間のパスが式 (3.9) を満足することである。製造されたチップ毎に対して、一定の環境（温度、圧力など）下でこの式の中の T_H、T_S、Δ_{ab}、t_a、t_b は制御できない値である。すなわち、ここでこれらは定数と見なされるべきである。製造後唯一積極的に制御できる値は $d_a - d_b$（PDE 遅延値の差）である。さらに、PDE のモデルによる PDE のレジスタ値は PDE の遅延値をコントロールしているから、PDE のレジスタ値 (離散制御信号値) は我々の制御対象となる。

図 3.2 にはタイミングテストに基づいて PDE 値を調整する例を示す。

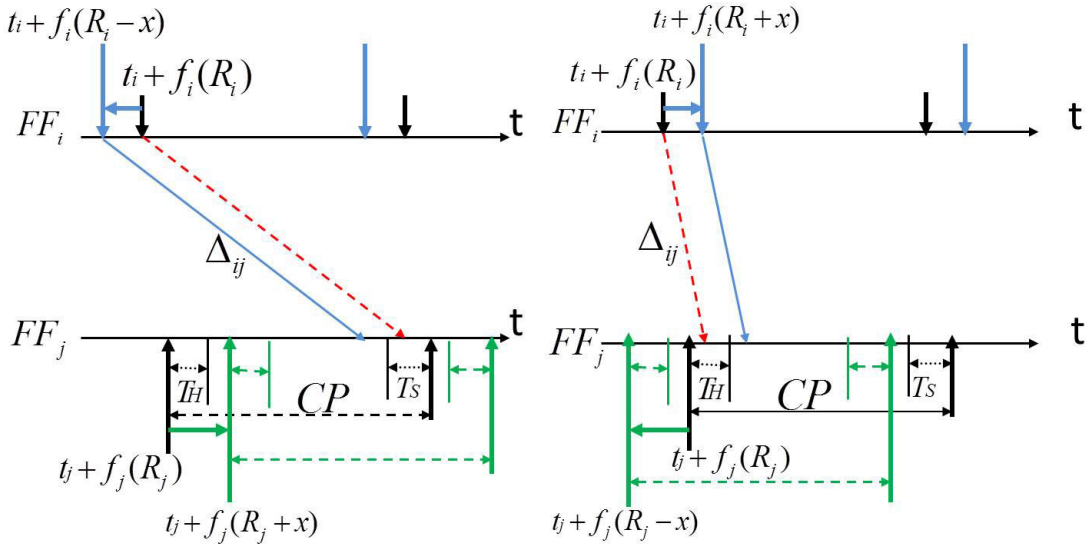

図 3.2: タイミングテストに基づく PDE 値調整の例

図 3.2 の左側は、FF_i から FF_j へのパスがセットアップ制約を違反する場合 (すなわち、$\exists P_i \in PATH$,such that $ffpair(P_i) = (FF_i, FF_j)$,and $testResult(P_i) = setupfail$) の PDE 調整の様子を示している。なお、$t_i$ と t_j は無調整時における FF_i,FF_j へのクロック信号到着時刻 (クロックソースからの遅延時間) であり、また FF_i,FF_j の現時点における離散制御信号値を R_i,R_j としている。

FF_i において $t_i + f_i(R_i)$ の時刻に入力値がラッチされて、出力値が新たにラッチされた値に切り替える ($t_1 + f_i(R_i)$ の黒矢印)。この FF_i 出力値更新に起因する信号伝搬が起こり、遅延時間 Δ_{ij} にて FF_j の入力端子に到着する (赤破線矢印)。しかしこの到着が FF_j のセットアップ制約時刻 $t_j + f_j(R_j) + CP - T_S$ より遅いためにセットアップ違反となっている。この違反を解消するためには、$f_i(R_i)$ を小さくし (R_i を R_{i-x} に減らし) て、FF_i からの信号が FF_j へより早く到着するようにする (青色矢印) か、$f_j(R_j)$ を大きくし (R_j を R_{j+x} 増やし) て、FF_j のセットアップ制約時刻を遅くする (緑色矢印) 必要がある。

図 3.2 の右側は、FF_i から FF_j へのパスがホールド制約を違反する場合 (すなわち、$\exists P_i \in PATH$,such that $ffpair(P_i) = (FF_i, FF_j)$,and $testResult(P_i) = holdfail$) の PDE 調整の様子を示している。なお、$t_i$ と t_j は無調整時における FF_i,FF_j へのクロック信号

到着時刻 (クロックソースからの遅延時間) であり、また FF_i,FF_j の現時点における離散制御信号値を R_i,R_j としている。

FF_i において $t_i + f_i(R_i)$ の時刻に入力値がラッチされて、出力値が新たにラッチされた値に切り替える ($t_1 + f_i(R_i)$ の黒矢印)。この FF_i 出力値更新に起因する信号伝搬が起こり、遅延時間 Δ_{ij} にて FF_j の入力端子に到着する (赤破線矢印)。しかしこの到着が FF_j のホールド制約時刻 $t_j + f_j(R_j) + T_H$ より早いためにホールド違反となっている。この違反を解消するためには、$f_i(R_i)$ を大きくし (R_i を R_{i+x} に増やし) て、FF_i からの信号が FF_j へより遅く到着するようにする (青色矢印) か、$f_j(R_j)$ を小さくし (R_j を R_{j-x} 減らし) て、FF_j のホールド制約時刻を早くする (緑色矢印) 必要がある。

要するに、具体的な遅延値を計測せずにタイミングテストの結果からタイミング違反の種類 (setupfail か holdfail か) を根拠として単に PDE の離散制御信号値を変えるだけで回路のタイミング違反を直すことが可能である。更に、式 (3.9) から見ると各 PDE の離散信号値より PDE の離散信号値の差が重要である。

本稿ではこの観点からアプローチする。従来の遅延量計測と数値計算による PDE 調整量決定と異なり、PDE 制御とタイミングテストを繰り返すことにより PDE 調整を行うアルゴリズムを提案する。

3.3 相対補正制約有向グラフ

タイミングテストの結果から、FF 間のクロック信号到着時刻に対する必要な補正を満たす PDE の離散制御信号値を見つけるために、相対補正制約有向グラフを定義する。

相対補正制約有向グラフ G は FF(PDE) を頂点とし、FF 間のタイミング制約を辺とする有向グラフである。提案したアルゴリズムはすべてこの相対補正制約有向グラフ $G(V, E)$ をめぐって展開される。

頂点集合は $V(G) = \{v_0, v_1, \ldots, v_{m-1}\}$ とする。頂点 v_i は FF_i と PDE_i を代表している。ですから、頂点 v_i が対応している FF_i と PDE_i の属性を共有している。頂点 v_i の離散制御信号値は実際に PDE_i の離散制御信号値である。

辺集合は $E(G) = \{(v_i, v_j) | v_i, v_j \in V(G)\}$ とする。辺の属性：

$color(v_i, v_j)$ は辺の色を示す。$color(v_i, v_j) \in \{red, green, colorless\}$。

$color(v_i, v_j) = red$ は FF_i に対して FF_j のクロック到着時刻を遅らせる必要があることを表している。

$color(v_i, v_j) = green$ は FF_i に対して FF_j のクロック到着時刻を遅らせる必要がないことを表している。

$color(v_i, v_j) = colorless$ は FF_i に対して FF_j のクロック到着時刻を遅らせる必要、或いは FF_j に対して FF_i のクロック到着時刻を遅らせる必要があるかどうかを判断できないことを表している。

$w(v_i, v_j)$ は各辺 (v_i, v_j) に対する、FF_i と FF_j 間のすべてのタイミング制約を満たすため最小必要な調整量の差 $R_j - R_i$ の下界を表示する辺の重みである。ここで R_j と R_i は

PDE の離散制御信号値であるが、前に述べたように頂点 v_i は FF_i と PDE_i を代表しているからである。毎回調整した後、グラフの全ての辺は $R_j - R_i \geq w(v_i, v_j)$ を満たさなければならない。

制約グラフ G は二つ要素で頂点の離散制御信号値を調整することをコントロールしている。辺の色は調整の方向をコントロールしている。一方、辺の重みは FF 間の最小必要な調整量の差として調整の量をコントロールしている。

ここで、次の補題 1 が成り立つ。

補題 1：相対補正制約グラフ G が辺重みについての正サイクルを持つ時、全ての制約 $R_j - R_i \geq w(v_i, v_j)$ を満足する離散制御信号値は存在しない。

証明：制約グラフ G に正サイクルが存在していると仮定する。この正サイクルを

$$v_0 v_1 \dots v_{i-1} v_i \dots v_n v_0$$

とする。辺 (v_i, v_j) の最小必要な差は w_i とする。各辺に関する不等式を全部例挙する。

$$R_1 - R_0 \geq w_0$$

$$R_2 - R_1 \geq w_1$$

$$\dots$$

$$R_n - R_{n-1} \geq w_{n-1}$$

$$R_0 - R_n \geq w_n$$

これらの不等式を足すと、$0 \geq \sum_{i=0}^{n} w_i$ になる。$\sum_{i=0}^{n} w_i > 0$ であるから、不成立である。以上の不等式組を満たす離散制御信号値が存在していない。証明ができた。

3.4 アルゴリズム

3.4.1 アルゴリズムの全体像

提案する PDE 調整アルゴリズムの流れを以下に示す。

step1: 制約グラフ G の初期化

step2: 回路のタイミングテスト (各 FF 間のセットアップ、ホールドタイミングテストを実施)、すべてのテストが成功すれば、終了する

step3: テストの結果により各辺を赤或いは緑に着色し、辺重み w を更新する

step4: 制約グラフ G の中に正重みサイクルがあるとき、調整不可能として、終了する

step5: 各辺の最小必要な差 w を満たす頂点の離散制御信号値を求める

step6: **step5** の結果を回路に適用して、**step2** へ

step5 には二つ方法に分けられる。「第一回の離散制御信号の設定」と「第二回以降の離散制御信号値の設定」である。

3.4.2 制約グラフ G の初期化

この部分の目的は相対補正制約グラフ G を構築することである。アルゴリズム「制約グラフ G の初期化」に示している。

Algorithm 制約グラフ G の初期化

1: $V(G) = \phi, E(G) = \phi$
2: **for all** $FF_i \in FF$ **do**
3: $V(G) = V(G) \cup \{v_i\}$
4: **end for**
5: **for all** $v_i, v_j \in V(G)$ **do**
6: **if** $\exists P, P \in PATH, ffpair(P) = (FF_i, FF_j) || ffpair(P) = (FF_j, FF_i)$ **then**
7: $E(G) = E(G) \cup \{(v_i, v_j), (v_j, v_i)\}$
8: **end if**
9: **end for**
10: **for all** $v_i \in V(G)$ **do**
11: $R_i = 0$
12: **end for**
13: **for all** (v_i, v_j) $inE(G)$ **do**
14: $w(v_i, v_j) = -\infty$
15: $color(v_i, v_j) = colorless$
16: **end for**

第 2 行から第 4 行までは一つの FF に対して一つの頂点を生成することを示す。第 5 行から第 9 行までは FF_i と FF_j の間に少なくても一つパスが存在すれば、この二つ FF の相対補正関係を表すために、辺 (v_i, v_j) と (v_j, v_i) を生成するべきことを示す。例えば、FF_i を始点とし FF_j を終点とするパスが一つだけ存在する場合、このパスがセットアップ制約条件を違反する場合、FF_i に対して FF_j のクロック到着時刻を遅らせる必要があることを表すために、辺の色の定義による $color(v_i, v_j) = red$ の前に辺 (v_i, v_j) が存在しなくてはいけない。このパスがホールド制約条件を違反する場合、FF_j に対して FF_i のクロック到着時刻を遅らせる必要があることを表すために、辺の色の定義による $color(v_j, v_i) = red$ の前に辺 (v_j, v_i) が存在しなくてはいけない。

第 10 行から第 16 行までは頂点の離散制御信号値と辺の重みと色を初期化する。各頂点の離散制御信号値を 0 と設定する。各辺の重み w を $-\infty$ と設定する。各辺の色を $colorless$ と設定する。

図 3.3 に初期化された一つの制約グラフ G の例を示している。丸い頂点の隣にある角括弧中の数字は頂点の離散制御信号値を示す。辺の隣にある丸括弧中の数字は辺の重みを示す。ここで、頂点離散制御信号値は全て 0 である。辺の重みは全て $-\infty$ である。辺 (v_0, v_3) と (v_3, v_0) の存在は FF_0 を始点とし FF_3 を終点とするパス或いは FF_3 を始点とし FF_0 を

終点とするパスが少なくとも一つ存在することを意味する。頂点 v_1 と v_3 の間に辺がないことは FF_0 を始点とし FF_3 を終点とするパス或いは FF_3 を始点とし FF_0 を終点とするパスが一つでも存在していないことを意味する。

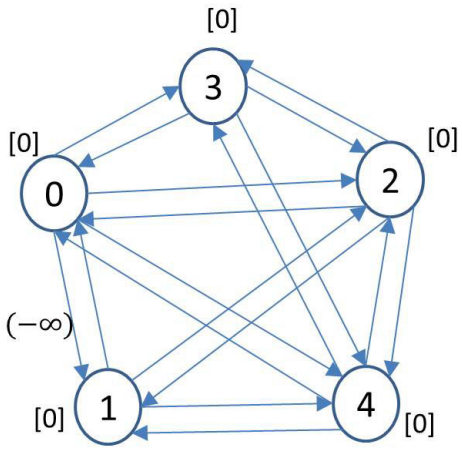

図 3.3: 制約グラフ G の初期化の例

3.4.3 タイミングテスト

この部分は回路のタイミングテストを行う。具体的なテスト方法はこの論文の議論の範囲外である。ここでタイミングテストからタイミングエラーの種類と対応する FF の情報を得られると仮定する。

3.4.4 辺の着色と w の更新

この部分に重要なことが二つある。一つはタイミングテストの結果によって各辺を赤い或いは緑に着色する。一つは辺の色と頂点の離散制御信号値によって辺重み w を更新する。アルゴリズム「辺の着色と w の更新」に示している。

第 1 行から第 12 行までは前者である。FF_i から FF_j への各パスの中にセットアップ制約条件を違反したパスが少なくても一つ存在すれば、或いは FF_j から FF_i への各パスの中にホールド制約条件を違反したパスが少なくても一つ存在すれば、FF_i に対して FF_j のクロック到着時刻を遅らせる必要があるから、$color(v_i, v_j) = red$ である。残された辺は全て $green$ とする。

第 13 行から第 17 行までは後者である。ここは第 N 回テストの後とすれば、第 N 回離散制御信号値を設定する前であるから、15 行目の R_i と R_j は実際に第 N-1 回テスト後に設定された離散制御信号値である。赤辺 (v_i, v_j) は第 N-1 回に設定された差 $R_j^{N-1} - R_i^{N-1}$ でタ

Algorithm 辺の着色と w の更新

1: **for all** $P \in PATH, ffpair(P) = (FF_i, FF_j)$ **do**
2: **if** $testresult(P) = setupfail$ **then**
3: $color(v_i, v_j) = red$
4: **else if** $testresult(P) = holdfail$ **then**
5: $color(v_j, v_i) = red$
6: **end if**
7: **end for**
8: **for all** $(v_i, v_j) \in E(G)$ **do**
9: **if** $color(v_i, v_j) \neq red$ **then**
10: $color(v_i, v_j) = green$
11: **end if**
12: **end for**
13: **for all** $(v_i, v_j) \in E(G)$ **do**
14: **if** $color(v_i, v_j) = red$ **then**
15: $w(v_i, v_j) = R_j - R_i + 1$
16: **end if**
17: **end for**

イミング誤りが起こったことを意味するので、差 $R_j^{N-1} - R_i^{N-1}$ が足りないことを表している。いわゆる、最小必要な差の下界が $R_j^{N-1} - R_i^{N-1}$ により大きい。だから、辺 (v_i, v_j) の最小必要な差の下界 $w(v_i, v_j)$ を $R_j^{N-1} - R_i^{N-1} + 1$ と設定する。緑辺 (v_i, v_j) は第 N-1 回に設定された差 $R_j^{N-1} - R_i^{N-1}$ で第 N 回のテストにタイミング誤りが起こってないことを意味する。$R_j^{N-1} - R_i^{N-1}$ が充分であることを表しているので、辺の最小必要な差の下界を変える必要はない。

図 3.4 と図 3.5 は制約グラフ G の辺に対する着色と w の更新の例を示している。図 3.4 は第一回テスト後の例を示す。図 3.5 はより一般的な例を示している。

図 3.4 の左側は初期化後の一つの制約グラフ G を示している。辺の色は全部 $colorless$ である。第 0 回テスト後と見なす。右側は第一回タイミングテスト後に着色され且つ w 更新された制約グラフ G の様子を示している。タイミングテストの結果によって辺 $(v_4, v_0), (v_0, v_3), (v_4, v_3), (v_1, v_2)$ の色が赤となる。他の辺は全て緑となる。ここで、新たな頂点離散制御信号値が算出されていないから、まだ第 0 回テスト後 (初期化後) の値を保持している。赤辺 (v_4, v_0) に対して最小必要な差の下界を更新する際、$w^1(v_4, v_0) = R_0^0 - R_4^0 + 1 = 0 - 0 + 1 = 1$ となる。他の赤辺の w は同じように更新される。緑辺の $w^1 = w^0 = -\infty$ として不変にする。

図 3.5 の左側は第一回離散制御信号値を設定した後の制約グラフの様子を示している。頂点の離散制御信号値は各辺の w を満たしている。しかし、もう一回のタイミングテストをしていないから、辺の色は全て $colorless$ である。右側は第二回タイミングテスト後に着色

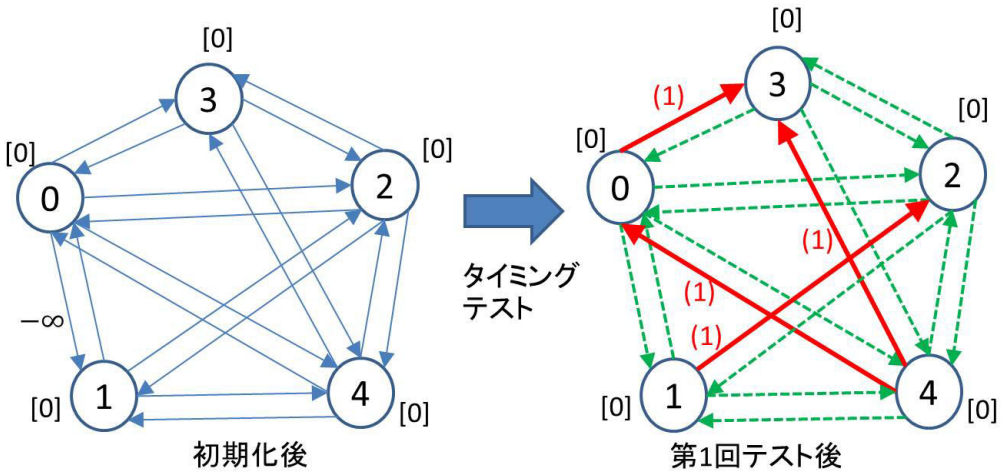

図 3.4: 辺の着色と w の更新の例 1

され且つ w 更新された制約グラフ G の様子を示している。第二回のタイミングテストの結果によって辺 $(v_4, v_0), (v_0, v_3)$ の色が赤となる。他の辺は全て緑となる。ここで、各頂点はまだ第一回テスト後に設定された離散制御信号値を保持している。赤辺 (v_4, v_0) に対して最小必要な差の下界を更新する際、$w^2(v_4, v_0) = R_0^1 - R_4^1 + 1 = 1 - 0 + 1 = 2$ となる。他の赤辺の w は同じように更新される。緑辺の w を不変にする。例えば、$w^2(v_4, v_3) = w^1(v_4, v_3) = 1$ となる。

3.4.5 第一回の離散制御信号値の設定

この部分は第一回テストして制約グラフの更新さらた各辺の重みを満たす頂点離散制御信号値範囲を算出して範囲内で頂点の離散制御信号値を求める。

以下はこの部分の全体像である。

step5.1 制約グラフ G の赤辺を全て取り出して、各頂点と一緒に新たなグラフ G_r を構成する

step5.2 グラフ G_r を対象に ASAP(as soon as possible) と ALAP(as late as possible) にて各頂点の可能な離散制御信号値の範囲を求める

step5.3 離散制御信号値を選択する必要がある頂点から一つを選び、その値を決める。もし離散制御信号値がまだ定まらない頂点が存在すれば、step5.2 へ

第一回テストして、step3 と step4 の後、制約グラフの赤辺の w は必ず 1 で、緑辺の w は必ず $-\infty$ である。

アルゴリズム [第一回の離散制御信号値の設定-1] は制約グラフ G の赤辺を全て取り出して、各頂点と一緒に新たなグラフ G_r を構成する。図 3.8 の (a) と (b) はこの過程の一つ例を示す。有向サイクルがない場合はグラフ G_r が DAG(有向非巡回グラフ) であるから、

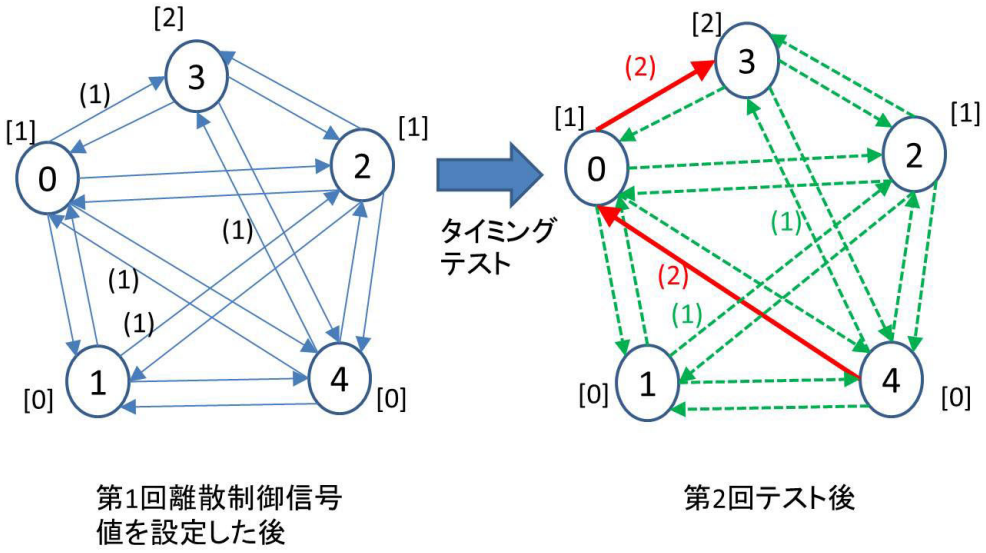

図 3.5: 辺の着色と w の更新の例 2

先行関係に基づいてグラフ G_r は図 3.6 のように n 部グラフになれる。この n 部グラフのトポロジカル順序に沿って右から左へ信号離散制御値を一つずつ増えていけば、各辺の最小必要な差を満たすことができる。しかし、図 3.6 の中の v のような“ 自由度 ”を持つ頂点が存在している。“ 自由度 ”の範囲内でどう移動しても先行関係を満たしている。“ 自由度 ”を持つ頂点の信号離散制御値をどう設定するのは問題になる。次のアルゴリズム「第一回の離散制御信号値の設定-2」と「第一回の離散制御信号値の設定-3」がこの問題を解決する。

Algorithm 第一回の離散制御信号値の設定-1

1: $V(G_r) = V(G), E(G_r) = \phi$
2: **for all** $(v_i, v_j) \in E(G)$ **do**
3: **if** $color(v_i, v_j) = red$ **then**
4: $E(G_r) = E(G_r) \cup \{(v_i, v_j)\}$
5: **end if**
6: **end for**

アルゴリズム「第一回の離散制御信号値の設定-2」は主に ASAP(as soon as possible) と ALAP(as late as possible) からなる。対象はグラフ G_r の離散制御信号値がまだ定まっていない頂点である。頂点の predecessor と successor の定義は図 3.7 で示す。第 2 行から第 7 行まで ASAP を行う。第 8 行から第 13 行まで ALAP を行う。実行した後に各頂点 v_i に対して、二つの値 R_i^{ASAP} と R_i^{ALAP} を持っている。$[R_i^{ASAP}, R_i^{ALAP}]$ の範囲内に頂点 v_i の離散制御信号値を選ぶ限り、各辺の最小必要な差を満たす。図 3.8 の (c) は一つの例を示

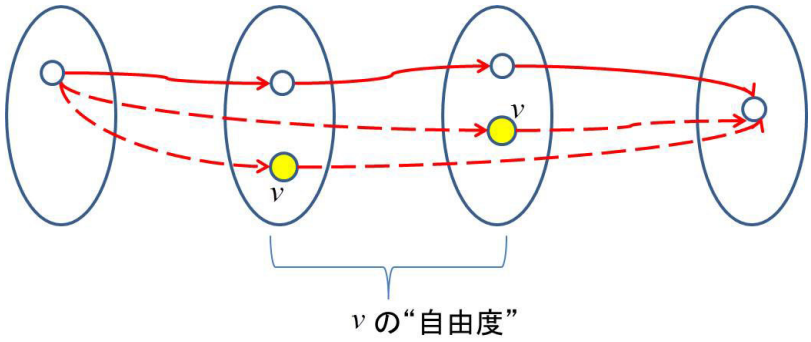

図 3.6: DAG(有向非巡回グラフ) の例

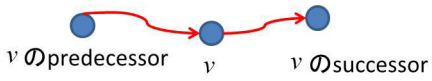

図 3.7: 頂点 v の predecessor と successor

している。頂点の隣に $[R_i^{ASAP}, R_i^{ALAP}]$ の形で算出された結果を表示する。例えば、頂点 v_1 の可能な離散制御信号値の範囲は $[0, 1]$ である。

$R_i^{ASAP} = R_i^{ALAP}$ を持つ頂点に対して、自由度がないから、頂点の離散制御信号値は R_i^{ASAP} で決まる。図 3.9 の (a) に頂点 v_4、v_0、v_3 はこのような頂点である。これらの頂点は“ 固定された頂点 ”とする。$R_i^{ASAP} \neq R_i^{ALAP}$ を持つ頂点に対して、範囲 $[R_i^{ASAP}, R_i^{ALAP}]$ 内に R_i を決めなければならない。まず、アルゴリズム「第一回の離散制御信号値の設定-3」第 1 行－第 4 行は図 3.10 に示すように“ 固定された頂点 ”と連結している緑辺の数が一番多い頂点を選ぶ。第 5 行－第 8 行には範囲内の各値に対する罰 (penalty) を計算することによって罰が最も小さい制御値を選ぶ。図 3.9 の (a) に示すように頂点 v_1 と v_2 の離散制御信号値を決める必要がある。しかし、頂点 v_1 に対して“ 固定された頂点 ”と連結している緑辺の数は 4 で、頂点 v_2 に対して“ 固定された頂点 ”と連結している緑辺の数は 6 であるから、まず頂点 v_2 を選んで各離散制御信号値の罰を計算する。

罰 (penalty) の計算方法を述べる前に、この段階の緑辺 (v_i, v_j) の意味を考えてみる。図 3.11 に示すように FF_i から FF_j への各パスがセットアップ制約条件を満たし (右)、且つ FF_j から FF_i への各パスがホールド制約条件を満たすから、辺 (v_i, v_j) が緑であることが成り立つ。図の中のように“ 余裕 ”が存在すると推測できるから、FF_j のクロック信号の到着時刻が早くなっても必ず誤るわけではない。この理由によって FF_j のクロック信号の到着時刻が早くなることに軽い罰を与える。逆に FF_j のクロック信号の到着時刻が遅くなることに重い罰を与える。

罰 (penalty) を計算する方法は図 3.12 に示す。

頂点 v_i' が固定された頂点と連結している緑辺 (v_i', v_j) と (v_j, v_i') を対象とする。頂点 v_i'

Algorithm 第一回の離散制御信号値の設定-2

for all $v_i \in V(G_r)$ whose discrete control value is not fixed **do**
  for all $v_i \in V(G_r)$ which have no predecessor **do**
    $R_i^{ASAP} = 0$
  end for
  for all $v_i \in V(G_r)$ whose predecessor v_j's number all fixed **do**
    $R_i^{ASAP} = \max_{v_j \prec v_i}\{R_j^{ASAP}\} + 1$
  end for
  for all $v_i \in V(G_r)$ which have no successor **do**
    $R_i^{ALAP} = \max_{v_j \in V(G_r)}\{R_j^{ASAP}\}$
  end for
  for all $v_i \in V(G_r)$ whose successor v_j's number all fixed **do**
    $R_i^{ALAP} = \max_{v_i \prec v_j}\{R_j^{ALAP}\} - 1$
  end for
end for

Algorithm 第一回の離散制御信号値の設定-3

for all $v_i \in V(G_r)$,$R_i^{ASAP} \neq R_i^{ALAP}$ **do**
  caculate numbers of green edge (v_i, v_j) and (v_j, v_i),$R_j^{ASAP} = R_j^{ALAP}$
end for
pickup $v_i^{'}$ who has the most green edges
for all $R \in [R_i^{'ASAP}, R_i^{'ALAP}]$ **do**
  caculate $penalty(R)$
end for
$R_i^{'} = R, penalty(R) = \min_{R_m \in [R_i^{'ASAP}, R_i^{'ALAP}]}\{penalty(R_m)\}$

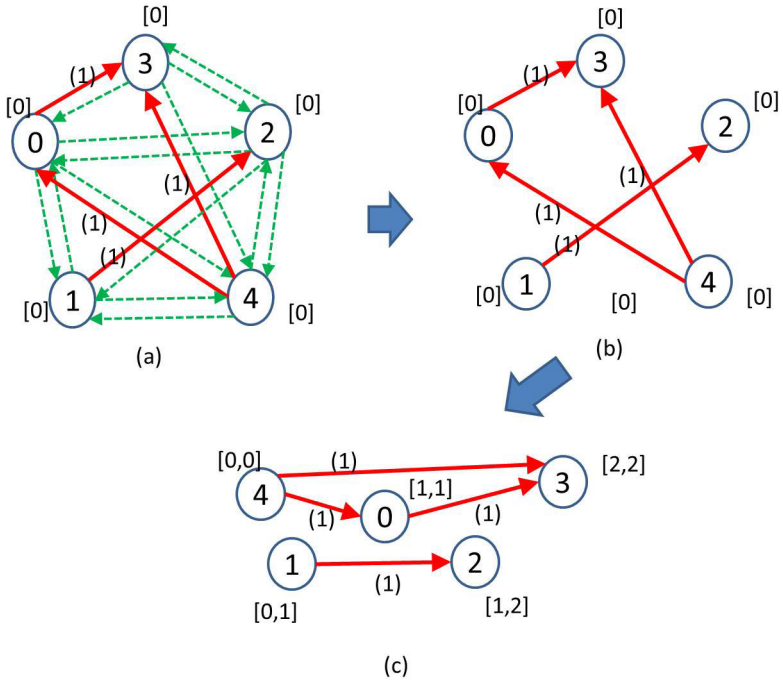

図 3.8: 第一回離散制御信号値の設定の例 1

の離散制御値が $R_i^{'}$ で設定される時の罰 $penalty(R_i^{'})$ を計算する。R_j は頂点 v_j の離散制御値である。α と β は係数である。$\alpha > 1, \beta < 1$。

緑辺 $(v_i^{'}, v_j)$ に対して、

$$penalty(R_i^{'}, (v_i^{'}, v_j)) = \begin{cases} \beta(R_i^{'} - R_j) & R_i^{'} > R_j \\ \alpha(R_j - R_i^{'}) & R_j > R_i^{'} \end{cases} \tag{3.10}$$

緑辺 $(v_j, v_i^{'})$ に対して、

$$penalty(R_i^{'}, (v_j, v_i^{'})) = \begin{cases} \alpha(R_i^{'} - R_j) & R_i^{'} > R_j \\ \beta(R_j - R_i^{'}) & R_j > R_i^{'} \end{cases} \tag{3.11}$$

だから、$penalty(R_i^{'}) = penalty(R_i^{'}, (v_i^{'}, v_j)) + penalty(R_i^{'}, (v_j, v_i^{'}))$ である。

図 3.9 の (b) に頂点 v_2 の各離散制御信号値に対する罰の計算結果を示している。1 の場合、罰は $2\alpha + 2\beta$ である。2 の場合、罰は $3\alpha + 3\beta$ である。だから、(c) のように頂点 v_2 の離散制御信号値を 1 と決める。

頂点 $v_i^{'}$ の離散制御信号値が“ 固定 ”された後、他の頂点の可能な離散制御信号値範囲も変わったので、もう一回「第一回の離散制御信号の設定-2」を行って、「第一回の離散制御信号の設定-3」で次の頂点の離散制御信号値を決める。全ての頂点の離散制御信号値が定まるまでこの手順を繰り返す。図 3.13 の (a) に頂点 v_2 の離散制御信号値を決めた後

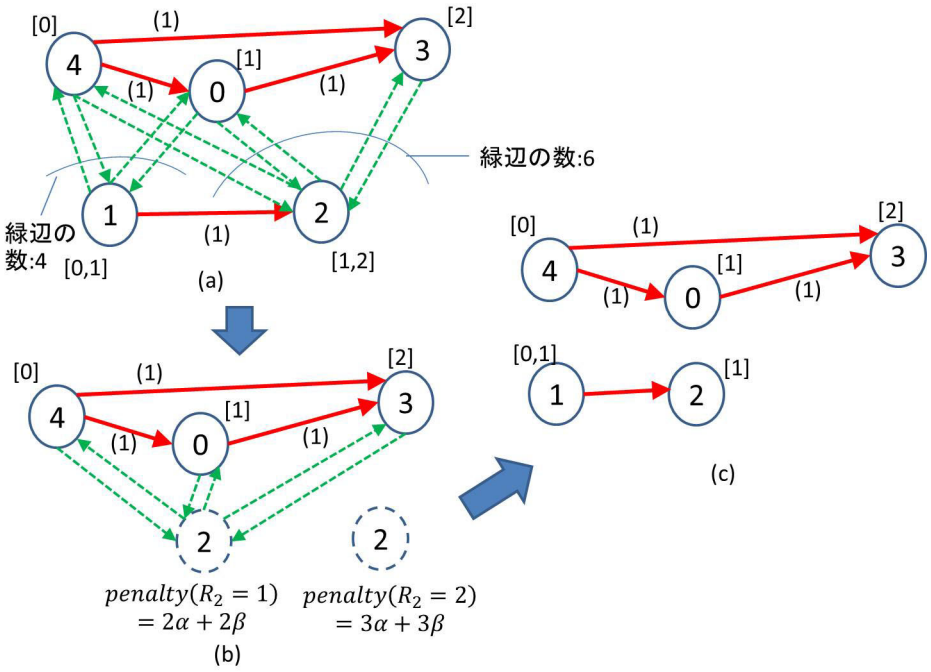

図 3.9: 第一回離散制御信号値の設定の例 2

に頂点 v_1 の離散制御信号値を決めなければならない。しかし、頂点 v_2 の離散制御信号値が“ 固定 ”されたから、頂点 v_1 と v_2 の先行関係によって v_1 の可能な離散制御信号値の範囲は $[0, 0]$ となる。すなわち、v_1 の離散制御信号値は 1 しか選択できない。ここまで、(c) が示すように各辺の重みを満たす第一回離散制御信号値の設定が完成した。

3.4.6 第二回以降の離散制御信号値の設定

この部分は第二回以降テストして制約グラフの更新さらた各辺の重みを満たす頂点離散制御信号値を求める。

第二回以降制約グラフの緑辺の w は $-\infty$ とは限らないので、第一回の方法が通用できない。各辺の重みを満たすために最長路問題になる。

アルゴリズム「第二回以降の離散制御信号値の設定」は Bellman-Ford アルゴリズム [10] に基づいて作られた単一始点最長路を解くものである。Bellman-Ford のアルゴリズムは辺の重みが負であってもよい、より一般的な場合の単一始点最短路問題を解くことができる。

第 1 行から第 6 行まで単一始点 s 及び他の頂点と連結する辺を作って新たなグラフ G_L を構成する。第 7 行から第 9 行まで各頂点の離散制御信号値を 0 と設定する。そして、第 10 行から第 16 行まで各辺に対して操作第 12 行と第 13 行を $|V(G_L) - 1|$ 回繰り返す。最後、全ての辺 (v_i, v_j) は $R_j - R_i \geq w(v_i, v_j)$ を満たす頂点の離散制御信号値を算出した。

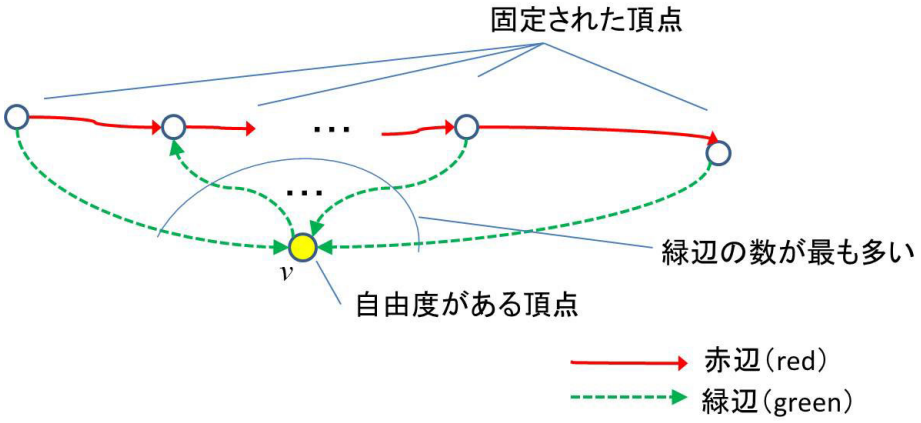

図 3.10: 自由度を持つ頂点の選び方

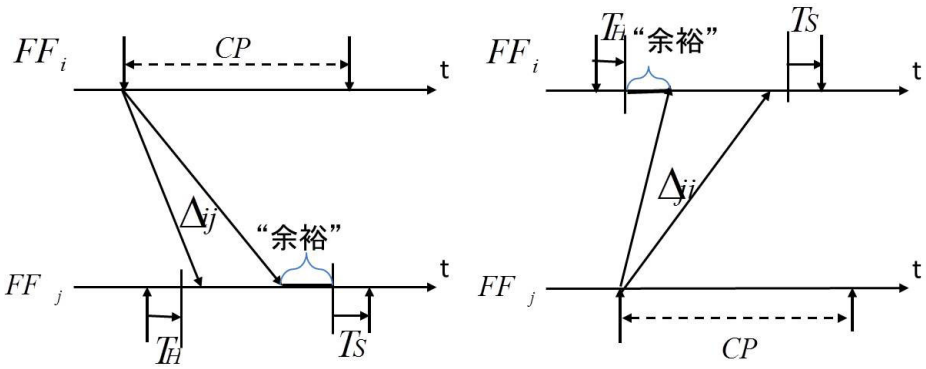

図 3.11: 緑辺の“ 余裕 ”

図 3.14 は第二回以降離散制御信号値の設定に関する一つの例を示している。(a) に始点 s 及び他の頂点と連結する辺 $(w = 0)$ が加えられる。(b) はアルゴリズム「第二回以降の離散制御信号値の設定」を行って算出された各頂点の値を示している。

3.4.7 アルゴリズムの有効範囲

提案するアルゴリズムについて、次の定理が成り立つ。

定理: すべての PDE について遅延量 $f_i(R_i)$ が制御信号に対して線形で、且つ傾きが等しい時、提案アルゴリズムによって構成される相対補正制約グラフ G が正サイクルを持つ時、全てのタイミング違反を解消する PDE 離散制御信号値は存在しない。

証明: 相対補正制約グラフ G において、w を更新するアルゴリズムによって $w(v_i, v_j) = w_i$ は、$R_j - R_i = w_i - 1$ を満たす“ ある ”R_j, R_i に対してタイミング誤りが起こったことを意味する。ここで PDE 遅延量の制御信号値に対する等傾き性により、$\forall R_x, R_y, R_m, R_n$,if

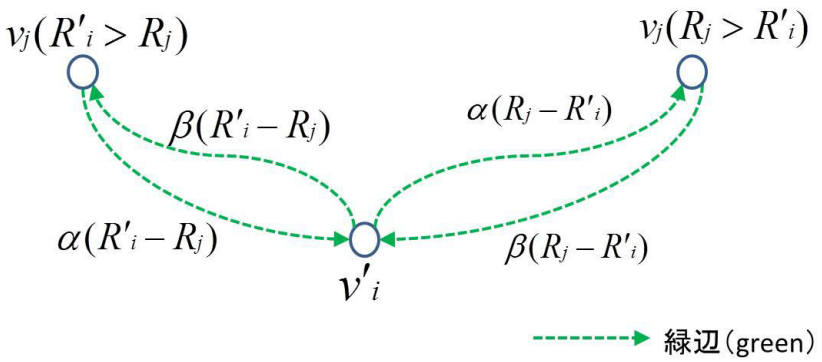

図 3.12: 罰 (penalty) の計算方法

$R_x - R_y = R_m - R_n$,then $f_x(R_x) - f_y(R_y) = f_m(R_m) - f_n(R_n)$ であるので、$R_j - R_i = w_i - 1$ を満たす“ すべての ”R_j, R_i に対してタイミング誤りが発生することが言える。すなわち、このタイミング誤りの解消には R_j, R_i の具体的な大きさに依存せず $R_j - R_i \geq w_i$ が必要である。このことと補題 1 より、定理が成り立つ。

一方、PDE の遅延量の制御信号値に対する傾きが PDE 毎に異なるとき、“ ある ”R_j, R_i、$R_j - R_i = w_i - 1$ におけるタイミング誤りは必ずしも他の R_j, R_i、$R_j - R_i = w_i - 1$ におけるタイミング誤りを意味しない。

図 3.15 は PDE の遅延量の制御信号値に対する傾きが PDE 毎に異なる時の 1 つ例を示している。$R_j - R_i = 0$ における PDE_i と PDE_j の制御信号値 $R_j = R_i = 3$ で FF_i と FF_j の間にタイミング誤りがあっても、実際の遅延値の差が (赤い線) 異なるので、$R_j = R_i = 6$ でタイミング誤りするとは限らない。

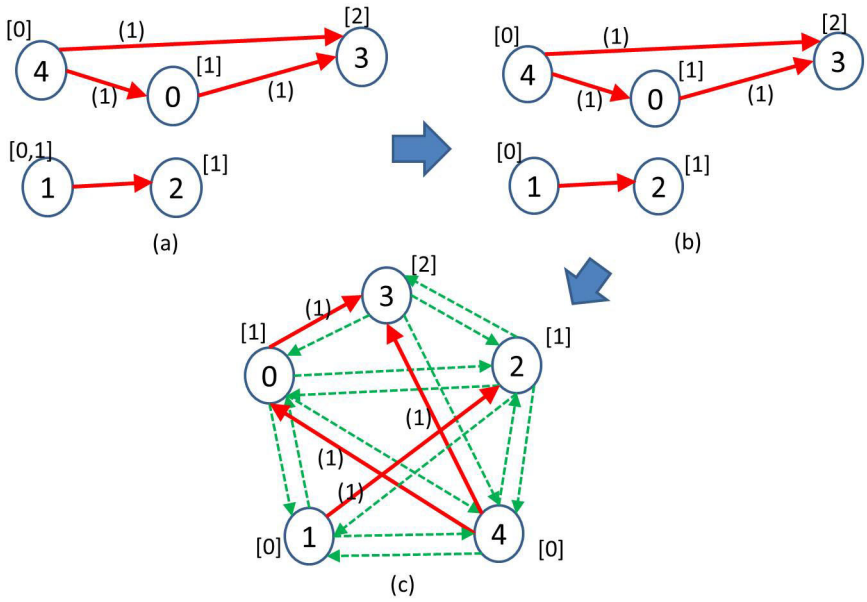

図 3.13: 第一回離散制御信号値の設定の例 3

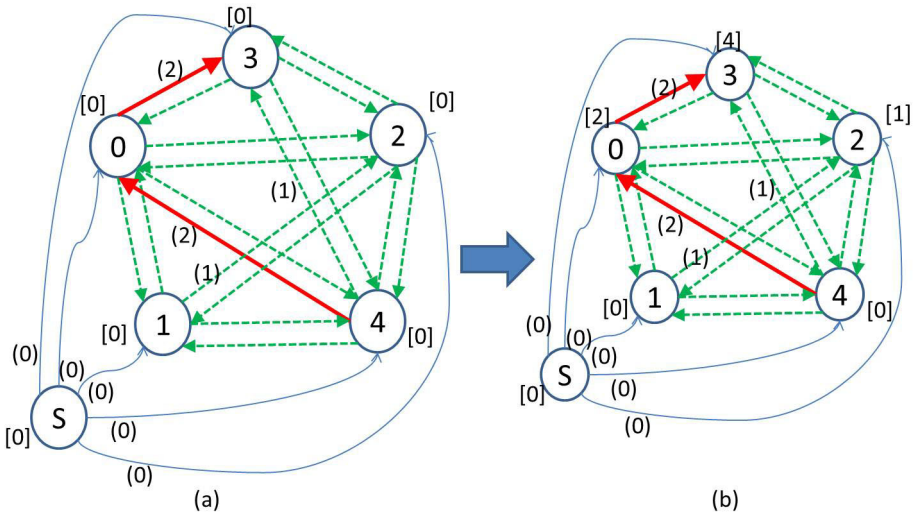

図 3.14: 第二回以降の離散制御信号値の設定の例

Algorithm 第二回以降の離散制御信号の設定

$V(G_L) = V(G) \cup \{s\}$
$E(G_L) = E(G)$
for all $v \in V(G)$ **do**
    $E(G_L) = E(G_L) \cup \{(s, v)\}$
    $w(s, v) = 0$
end for
for all $v_i \in V(G_L)$ **do**
    $R_i = 0$
end for
for $i = 1$ to $|V(G_L) - 1|$ **do**
    for all $(v_i, v_j) \in E(G_L)$ **do**
        if $R_j < R_i + w(v_i, v_j)$ **then**
            $R_j = R_i + w(v_i, v_j)$
        end if
    end for
end for

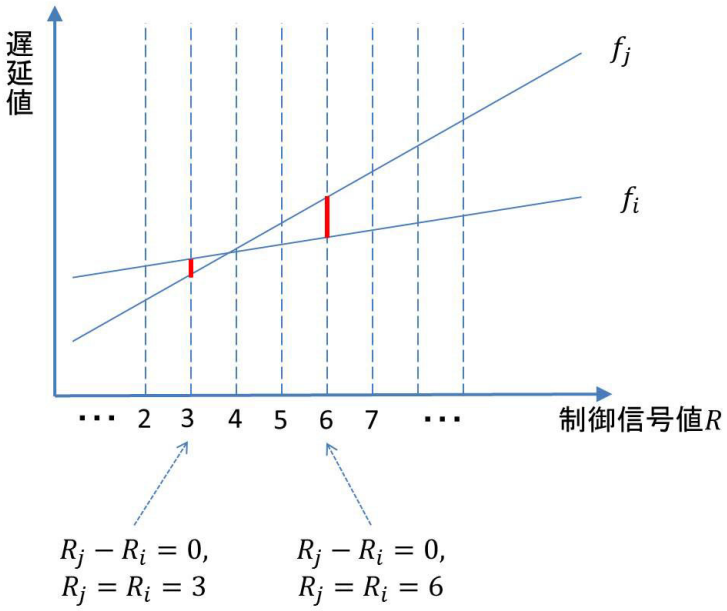

図 3.15: 傾きが異なるときの 1 つ例

第4章　実験結果と分析

本稿で提案するアルゴリズムを評価するために、いくつかの回路を対象にc言語とPerlを使ってシミュレーション実験を行った。実験PC環境は2.67GHz×4Intel(R)Core(TM),4GBメモリーである

4.1　調整性能に関する実験

実験は図4.1の流れで行われる。

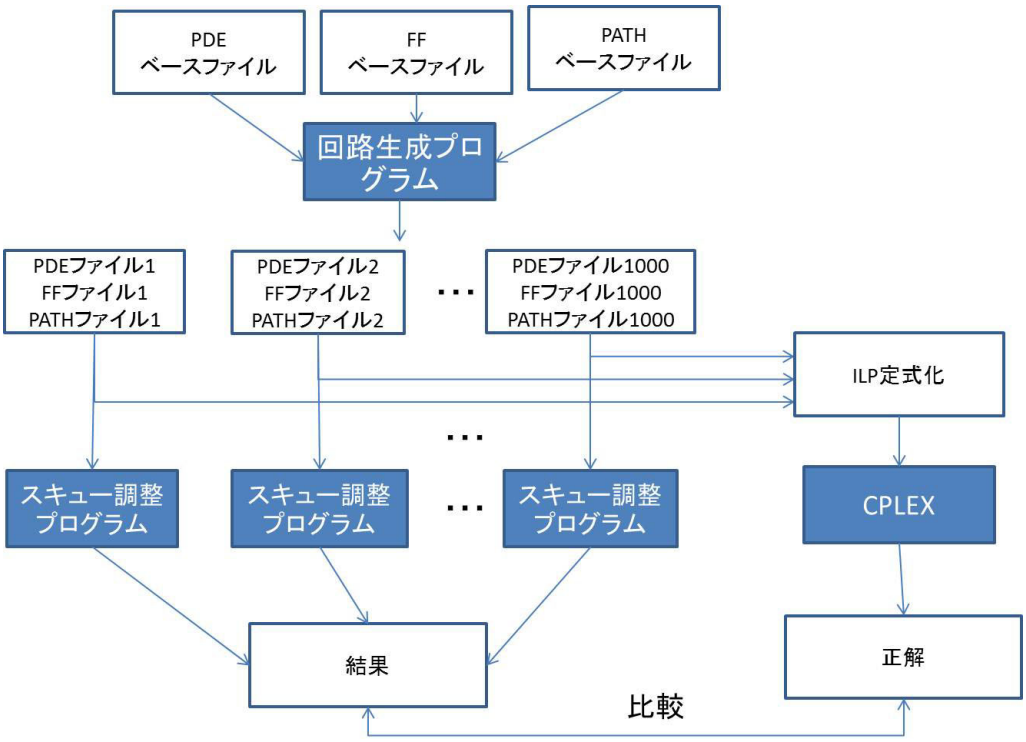

図 4.1: 実験の流れ

ここでPDEの遅延関数は$f(R) = a + bR + cR^2$と設定する。a,b,cは多項式の係数である。これらの係数によってPDEの遅延値が算出できる。

表 4.1: 回路情報

回路	回路 1	回路 2(s208.1)	回路 3(s1423)
FF 数 (PDE 数)	5	8	74
PATH 数	25	144	71980
クロック周期 (ns)	2	0.5	5
セットアップとホールド時間 (ns)	0.1	0.01	0.01

表 4.2: 実験結果

結果の分類	回路 1	回路 2	回路 3-a	回路 3-b
ゼロ調整動作	225(22.5%)	215(21.5%)	26(2.6%)	437(43.7%)
調整成功	328(32.8%)	495(49.5%)	455(45.5%)	187(18.7%)
調整放棄失敗	60(6%)	72(7.2%)	96(9.6%)	53(5.3%)
調整放棄成功	387(38.7%)	218(21.8%)	423(42.3%)	323(32.3%)

各 PDE の各係数、各 FF のクロック信号到着時刻、各パスの遅延値が各自の正規分布 $N(\mu, \sigma^2)$ に従うものとする。

一つの回路の情報は以下の三つのファイルに記録される。

PDE ベースファイル：各 PDE の遅延関数の係数の正規分布情報を記録する。

FF ベースファイル：各 FF のクロック信号到着時刻の正規分布情報を記録する

PATH ベースファイル：各パスの遅延値の正規分布情報を記録する。

一つの回路に対して、遅延ばらつきを加えて 1000 個の回路を生成する。ばらつきを加えられた回路の情報が以下のファイルに記録される。

PDE ファイル：各 PDE の遅延関数の係数を記録する。

FF ファイル：各 FF のクロック信号到着時刻を記録する。

PATH ファイル：各パスの遅延値を記録する。

そして、これらの 1000 個回路それぞれに対して、スキュー調整アルゴリズムを実行する。なお、結果を“ 正解 ”と比較するために、ILP を用いた PDE 制御信号値計算を併せて行った。“ 正解 ”を計算するツールは IBM　CPLEX　Interactive Optimizer 12.1.0 である。

表 4.1 は回路の情報を表している。回路 1 の構造と各伝搬遅延値と到着時刻は実験的に自分で設計されたものである。回路 2 と回路 3 それぞれの構造は ISCAS'89benchmark の s208.1 と s1423 と一致である。回路 2 と回路 3 の各伝搬遅延値は NangateOpenCellLibrary によって Primetime で算出された。算出された各値は各パスの遅延値の正規分布の平均値とする。この上に各正規分布の標準偏差値を設定することによってばらつきを付ける。

表 4.2 は、PDE 調整の成功数、失敗数を表している。「ゼロ調整動作」は PDE 調整なし (制御信号値がすべて 0) で動作した回路数、「調整成功」は、PDE 調整にて正しく動作した回路数、「調整放棄失敗」は、回路が正しく動作する PDE 調整が存在するにもかかわらず、アルゴリズムがそれを見つけられなかった回路数、「調整放棄成功」は、回路が正しく動作する PDE 調整が存在しない回路数を、それぞれ表している。調整放棄の原因はグラフに正重みサイクルが存在することである。ここで、回路 3-a と回路 3-b の各正規分布の標準偏差値の設定方法、すなわち、ばらつきの付け方だけが違う。

図 4.2 は、各回路の調整成功と調整放棄成功と調整放棄失敗に応じるテスト回数をグラフしたものである。

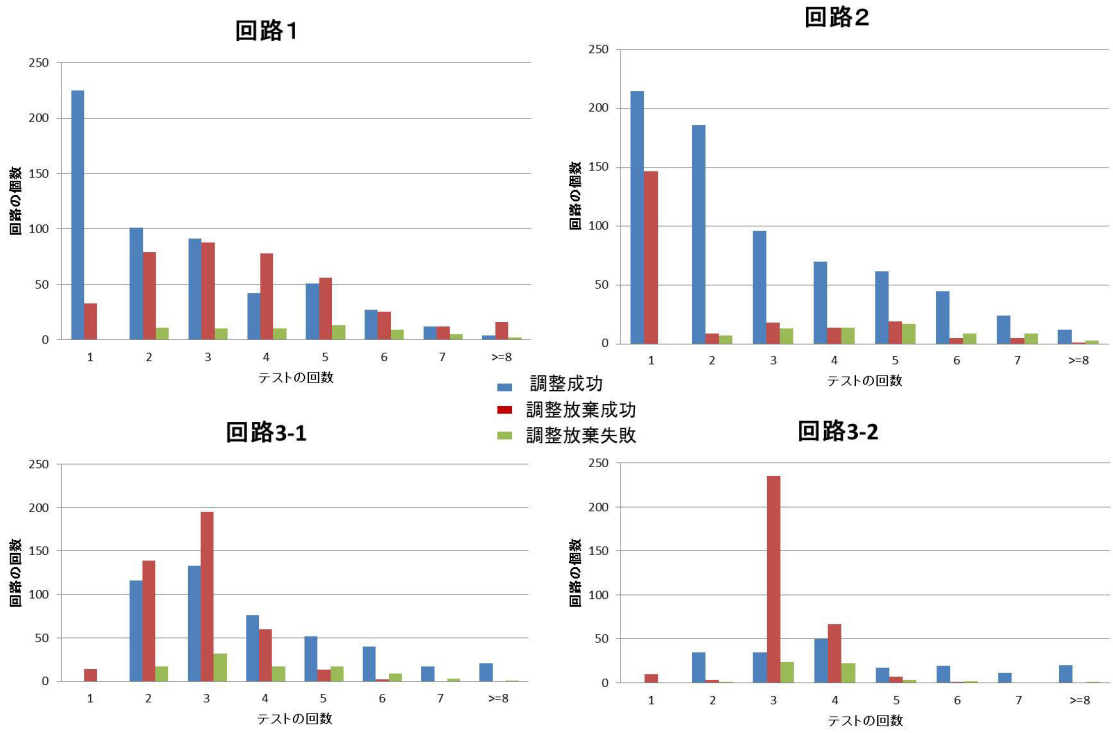

図 4.2: テスト回数による統計

4.2 最小タイミングマージンの上界に関する実験

回路の FF 間のタイミングマージンは図 4.3 に示す。最小タイミングマージンの上界とは各遅延量を使って理論上の FF 間の各マージンから選ばれた最小値である。

「調整成功」回路と「調整放棄失敗」回路の ILP 定式化により各回路の最小マージンの上界を算出した。図 4.4 は、回路の最小タイミングマージン毎の調整成功数と調整放棄

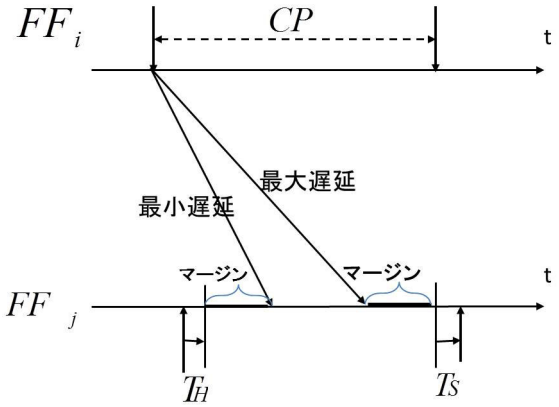

図 4.3: 回路のタイミングマージン

表 4.3: 回路 1 における PDE 遅延曲線に関する実験結果

PDE 遅延曲線分類	typeA	typeB	typeC	typeD
ゼロ調整動作	225(22.5%)	225(22.5%)	33(3.3%)	225(22.5%)
調整成功	305(30.5%)	335(33.5%)	508(50.8%)	251(25.1%)
調整放棄失敗	0(0%)	35(3.5%)	0(0%)	124(12.4%)
調整放棄成功	470(47.0%)	405(40.5%)	459(45.9%)	400(40.0%)

失敗数の分布をグラフしたものである。結果から見ると、最小マージンの上界が小さい回路は調整放棄失敗の確率が高い。

4.3 PDE の遅延曲線に関する実験

実験は図 4.5 の流れで行われる。

「調整放棄失敗」の原因を見つけるために、PDE の遅延曲線に関する実験を行った。回路 1 からばらつきを加えられた 1000 個回路に対して PDE の遅延曲線をある規律に従わせる。図 4.6 の中 typeA のように全ての PDE の遅延曲線が同じく直線に従う。typeB のように PDE の遅延曲線が同じく非線形である。typeC のように PDE の遅延曲線が線形で、傾きが同じな平行線である。typeD のように PDE の遅延曲線が線形であるが、傾きが違う。表 4.3 は、PDE 遅延曲線に関する実験結果を示している。

同じ方法で回路 2 に対して PDE 遅延曲線に関する実験を行った。結果は表 4.4 と図 4.7 に示している。

PDE の遅延曲線に関する実験結果から、typeB と typeD の場合は「調整放棄失敗」の回路が現れた。連続値で考えたら、各辺に対して元々$d_j - d_i \geq w'_{ij}$ を満たす PDE の遅延値を

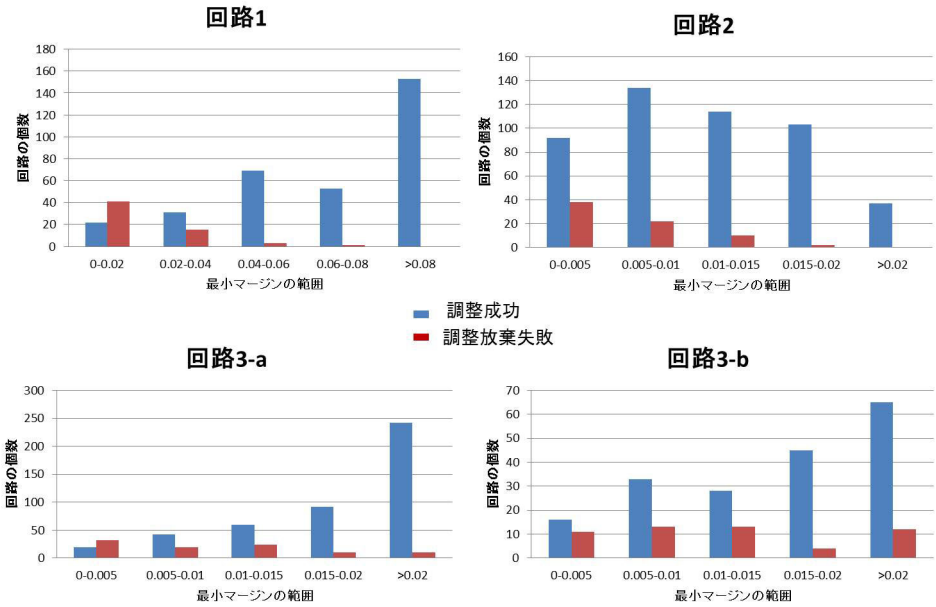

図 4.4: 最小タイミングマージンの上界による統計

表 4.4: 回路 2 における PDE 遅延曲線に関する実験結果

PDE 遅延曲線分類	typeA	typeB	typeC	typeD
ゼロ調整動作	217(21.7%)	217(21.7%)	147(14.7%)	217(21.7%)
調整成功	522(52.2%)	541(54.1%)	595(59.5%)	428(42.8%)
調整放棄失敗	0(0%)	23(2.3%)	0(0%)	132(13.2%)
調整放棄成功	261(26.1%)	219(21.9%)	258(25.8%)	223(22.3.0%)

見つかる。ここで $d_j, d_i, w_{ij}^{'} \in R$ である。本手法ではこの問題から離散的な問題に転換する際に一つ前提を設置した。PDE 遅延曲線の各区間に傾きが同じという前提である。所謂 $\forall R_x, R_y, R_m, R_n$,if $\quad |R_x - R_y| = |R_m - R_n|$,then $\quad |f_x(R_x) - f_y(R_y)| = |f_m(R_m) - f_n(R_n)|$ である。隣接している離散制御信号値に対応する遅延値の差 $\delta = |f_x(R_x + 1) - f_x(R_x)|$ が一定であれば、この値は調整最小単位値と見られる。だから、$d_i = R_i\delta, d_j = R_j\delta, w_{ij}^{'} = w_{ij}\delta, R_i, R_j, w_{ij} \in N$ によって $R_j - R_i \geq w_{ij}$ になる。typeB と typeD の場合ではこの前提がなくなったから、“ 正解 ”が存在しているのに、提案手法がある区間で正重みサイクルを発見したので、すぐ止まってしまった。図 4.8 には一つ「調整放棄失敗」の例の正重み正サイクルを示している。丸は頂点である。丸の中にある数字は頂点の番号である。長方形の中には離散制御信号値 (遅延値) を表す。提案手法によって各辺の最小必要な整数差を満たす離散制御信号値が存在していない。しかし、辺 (v_4, v_1) を見れば、$R_4 = 10, R_1 = 10$

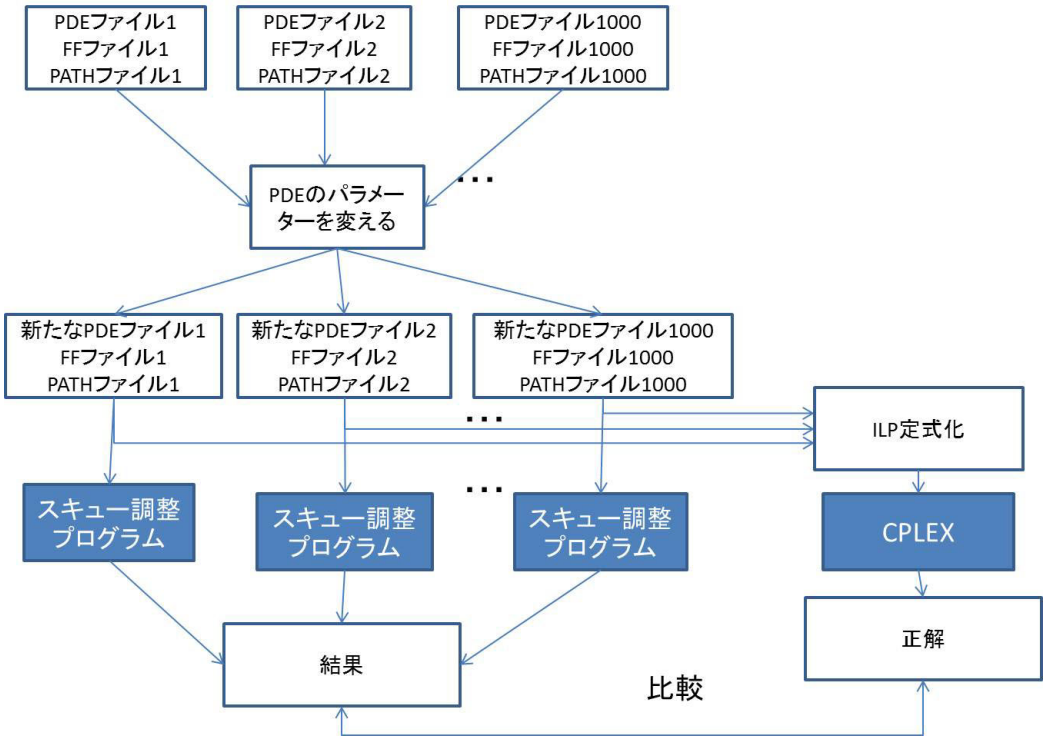

図 4.5: PDE の遅延曲線に関する実験の流れ

で辺の最小必要な整数差を満たしていないが、辺の最小必要な実数差を満たしている。整数で辺の間にある本当な最小必要な差を見積もる考えは欠陥があるので、この考えを基に提案された手法は無論に欠陥が存在する。

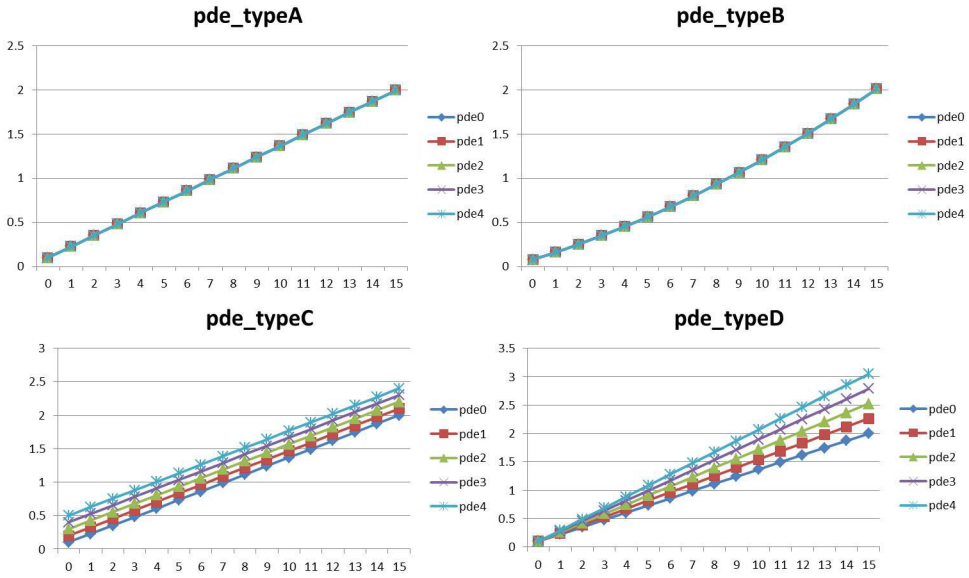

図 4.6: 回路 1 における PDE 遅延曲線の種類

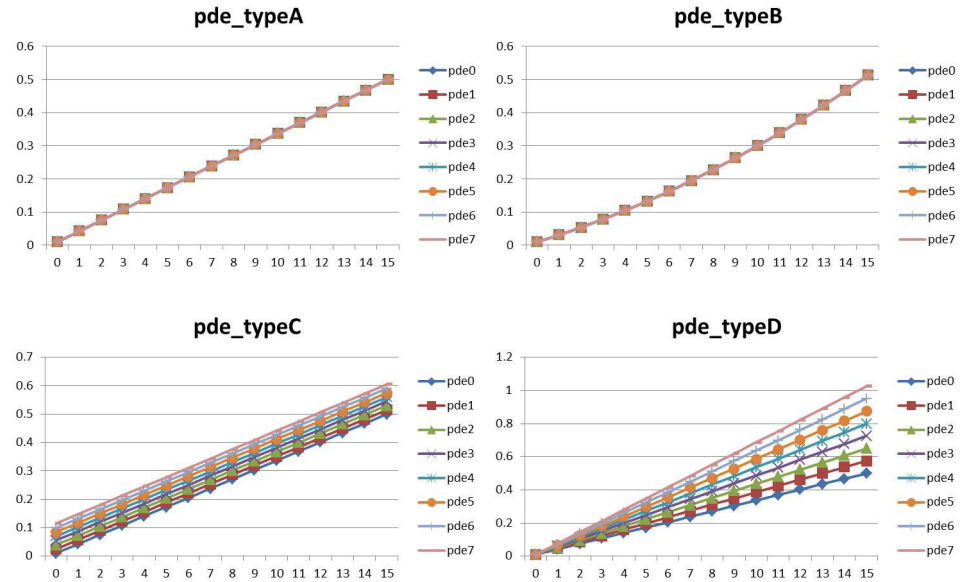

図 4.7: 回路 2 における PDE 遅延曲線の種類

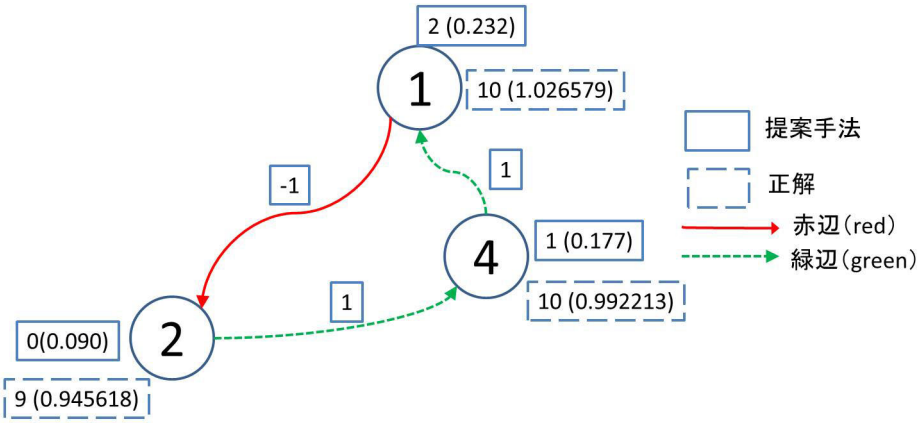

図 4.8: 正重みサイクルの一つ例

第5章　まとめと今後の課題

本稿では、従来の遅延量計測と数値計算によるPDE調整量決定と異なり、FF(PDE)を頂点とし、FF間のタイミング制約を辺とする有向重み付きグラフ(相対補正制約有向グラフ)を定義する。辺の重みは二つFF間のスキューを調整する最小必要な調整量の差とする。具体的な遅延情報を計測せずにタイミングテストから得られたタイミングエラー種類と対応するFFの情報を用いて、制約グラフの各辺の更新された重みを満たすPDEの離散制御信号値を求める。所謂、タイミングテストとPDEの離散制御信号値調整を繰り返すことによりタイミングエラーの問題を解決する発見的なアルゴリズムを検討、提案した。

実験結果により、タイミング違反が存在する回路の歩留まりを改善する有効性を確認した。PDE遅延曲線に関する実験により、アルゴリズムの不足点を判明し、検討した。PDE遅延量の制御信号値に対する傾きのばらつきに対する調整性能の向上が一つの課題として残されている。

実用的には回路のタイミング違反を解消する上に環境の変動(温度、電圧など)に強い、いわゆる回路のタイミングマージンを最大化する調整手法を検討する必要がある。

今のモデルの中に各FFのクロック信号線に一つのDPEを挿入すると仮定しているが、実際に回路の規模を大きくしないためにPDEの挿入コストを削減するのは今後の一つ課題である。

タイミングテストについてすべてのパスのタイミングテスト結果を得られるとは限らないので、調整性能を保証する上にテストコストの削減が今後の課題になる。

謝辞

本研究を進めるにあたり, 金子峰雄教授より暖かいご指導を受け賜りました．ここに深く感謝の意を表します．また多くの助言を頂いた岩垣剛助教, 研究員井上恵介氏, 研究室の皆様にも深く感謝いたします.

参考文献

[1] L.Scheffer, S.Nassif, A.Strojwas, B.Koenemann, and N.S.Nagaraj,“ Design for Manufacturing in the sub-65nm Era ”,Tutorial in DAC 2005, 2005.

[2] J.singh, S.Sapatnekar,“ Statistical Timing Analysis with Corrected Non-Gaussian Parameters using Independent Component Analysis ”,Proc. of DAC 2006,pp.155-160,2006

[3] R.Ginosar,Y.Elboim,and A.Kolodny,“ A clock tuning circuit for system-on-chip ”,In Proceedings of the second ACiD-WG workshop of the european commission's fifth framework programme,2002

[4] Hashizume Y, Takashima Y, Nakamura Y,“ A novel clock deskew method by linear programming ”, Proc. of Midwest Symposium on Circuits and Systems, pp.1261-1264,2007

[5] 大谷　直毅, 橋爪　裕子, 高島　康裕, 中村　祐一,“ 統計的推定を用いたクロックデスキューに対する一手法 ”, 信学技報．Vld2006-126,pp．43-48,2006

[6] 橋爪　裕子, 高島　康裕, 中村　祐一,“ クロック信号におけるばらつきが測定不要なデスキュー手法 ”, 信学技報．CAS2007-95,pp．7-12,2008

[7] Eiichi Takahashi, Yuji Kasai, Masahiro Murakawa,and Tetsuya Higuchi,“ Post-Fabrication Clock-Timing Adjustment Using Genetic Algorithms ”,IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 39, NO. 4, APRIL 2004

[8] Nagaraj K, Kundu S,“ An Automatic Post Silicon Clock Tuning Systemfor Improving System Performance based on Tester Measurements ”,IEEE International Test Conference,pp.1-8,2008

[9] Maymandi Nejad M, Sachdev M,“ A digitally Programmable delay element:design and analysis ”,Very Large Scale Integration (VLSI) Systems, IEEE Transactions,VOL.11,NO.5,OCTOBER 2003,pp.871 - 878

[10] Thomas H. Cormen,Charles E. Leiserson,Ronald L. Rivest,“ Introduction to Algorithms ”,The MIT Press,pp.514 - 536