修 士 論 文

タイミングスキュー調整可能データパスのための設計制約と合成法

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

手原　亮

2010 年 3 月

修 士 論 文

タイミングスキュー調整可能データパスのための設計制約と合成法

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　日比野 靖 教授
審査委員　田中 清史 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

0810039 手原　亮

提出年月: 2010 年 2 月

概 要

集積回路の微細化，動作速度の向上に伴い，製造ばらつきによる回路内の信号伝搬遅延のばらつきが相対的に大きくなりつつある．タイミングマージンを十分に取り回路の正常動作を保証する従来の回路設計手法では，回路性能の向上が困難である．この問題に対して，信号伝搬遅延量に応じてレジスタ書き込みタイミングに意図的にスキューを導入し，回路の正常動作を保証する手法が提案されている．しかし，高い歩留まりの達成は与えられるデータパスに強く依存している．本研究では，タイミングスキュー調節を前提に，回路の持つポテンシャルを最大限に引き出し，高い歩留まりを達成することが可能なデータパスの合成理論と手法の開発を目標とする．本稿では，この目標にアプローチする準備として，タイミングスキュー調整成功率を導入すると共に，その計算手法を提案する．次いで，タイミングスキュー調整を考慮した演算器割当て問題に取り組む．タイミングスキュー調整が不可能な演算器割当て条件を明らかにし，その条件を基に割当て手法を提案する．なお，タイミングスキュー調整を考慮したレジスタ割当ては今後の課題である．

目 次

第1章　はじめに

集積回路の歴史は，半導体製造技術の進歩による回路の微細化，動作速度の向上の歴史と言える．従来では，ゲートのスイッチング遅延に対して配線遅延は無視できるほど小さかったが，こうした微細化と速度向上により配線遅延が無視できなくなってきている．そのため現在主流の回路方式である同期式回路において，クロック信号を各レジスタに位相差なく分配することが困難になっている．また，製造ばらつきによる回路内の信号伝搬遅延のばらつきが相対的に大きくなりつつある [7]．これらの要因により，タイミングエラーによる歩留まりの低下が問題となってきている．この問題に対して，これまでタイミングマージンを十分に取ることで対応してきたが，一方で過剰なタイミングマージンは回路性能の低下を招くことになる．より高性能な集積回路が求められる中，従来手法では所望の回路性能を達成することが困難になりつつある．近年，統計的遅延解析を導入して遅延見積もりの精度を上げ，歩留まりと性能のトレードオフを考慮してタイミングマージンを適切に設定する手法が提案されているが [4][10][9]，正常動作をマージンに頼る点は変わらず，個別チップの性能を十分に引き出せているとは言えない．こうしたアプローチとは別に，回路製造後にチップ毎に回路を調整するアプローチもある．クロック・スキューに起因するタイミングエラーを対象とし，クロック・スキュー解消を目的とするデススキュー手法 [8] や，逆に信号伝搬遅延量に応じてレジスタ書き込みタイミングに意図的にスキューを導入し，タイミングエラーを回避する手法などが提案されている [6]．しかし，高い歩留まりの達成は与えられるデータパス (回路の構造記述と制御記述) に強く依存している．本研究では，タイミングスキュー調節を前提に，回路の持つポテンシャルを最大限に引き出し，高い歩留まりを達成することが可能なデータパスの合成理論と手法の開発を目標とする．特に本稿では，製造ばらつきを対象として，タイミングスキュー調整を考慮したデータパス合成の一つである演算器割当てについて考える．

本稿は以下のように構成される．第 2 章では既存手法であるタイミングスキュー調整について説明する．第 3 章では本研究で性能評価の 1 つとして導入するタイミングスキュー調整成功率について説明し，その計算手法を提案する．第 4 章ではタイミングスキュー調整を効果的に行うための演算器割当て条件を示す．第 5 章ではその条件を基に演算器割当て手法を提案する．第 6 章でまとめと今後の課題について述べる．

第2章　タイミングスキュー調整可能データパス

本章では，回路の構造記述と制御記述であるRTLデータパス回路を対象に，既に提案されているタイミングスキュー調整手法について説明する．

2.1　RTLデータパス回路

RTLデータパス回路とは，記憶素子であるレジスタ，信号を切り替えるMUX，種々の計算を行う演算器から構成される．このデータパス回路には，コントローラからクロック信号に同期したレジスタの書き込み制御信号やMUXの切り替え制御信号などが入力されている．具体例として図2.2に示すアルゴリズムを実行するデータパス回路を図2.1に示す．このデータパス回路は，回路の動作を記述したアルゴリズム記述より，スケジューリング，演算器割当て，レジスタ割当てを行うことにより生成することができる．アルゴリズム記述とは，所望の機能，システムをアルゴリズムで表現したものであり，データの流れを表現するDFG(Data Flow Graph)と演算順序などの制御を表現するCFG(Control Flow Graph)で表現され，両者を表現できるCDFG(Cntrol Data Flow Graph)が度々利用される．以降は，図2.2に示すDFGを用いて解説を行う．DFGとは，演算とその依存関係を表す有向辺を持つ有向グラフ$G = (V, E)$である．頂点集合$V = (\mathcal{O} \cup \mathcal{Q})$は，演算集合$\mathcal{O}$と外部入出力を表すダミー演算集合$\mathcal{Q}$の和集合から成る．図2.2では，$E = (A + B) * (C + D)$を表現している．

2.1.1　スケジューリング

スケジューリングとは，ハードウェア資源や総ステップ数などの時間制約を考慮し，具体的にどのコントロールステップで各演算を実行するかを決定することである．コントロールステップとは，離散的な時間のことである．つまりスケジューリングとは，演算集合$\mathcal{O}$から整数$\mathbb{Z}$への写像$\sigma : \mathcal{O} \mapsto \mathbb{Z}$である．図2.3に，スケジューリングの例を示す.(a)では，コントロールステップ:0に2つの加算演算が割り当てられている．また乗算は2ステップ演算としてコントロールステップ:1,2に割り当てられている．この時，総コントロールステップ数は3ステップである．最小演算器数は，加算器数2，乗算器数1である．また，最小レジスタ数は4である．(b)では，コントロールステップ:0,1に各加算

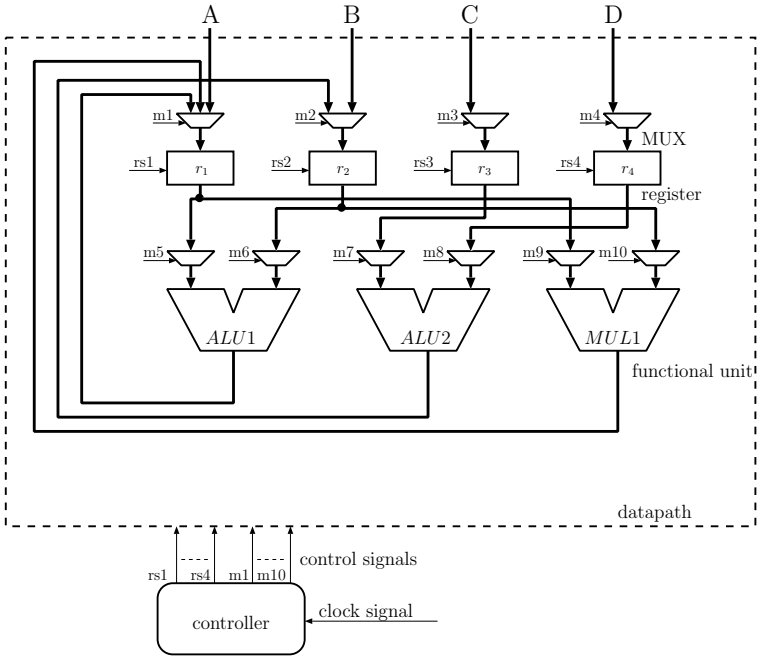

図 2.1: RTL 回路の例

演算が割り当てられている．また乗算は 2 ステップ演算としてコントロールステップ:3,4 に割り当てられている．この時，総コントロールステップ数は 4 ステップである．最小演算器数は，加算器数 1，乗算器数 1 である．また，最小レジスタ数は 3 である．このようにスケジューリングは様々なバリエーションが考えられ，例のように実行時間と資源数がトレードオフの関係となる場合がある．スケジューリングアルゴリズムは，制約の与え方により資源制約スケジューリングと時間制約スケジューリングに分類することができる．前者に関しては，使用可能な資源数の上限が与えられたもとでコントロールステップ数が最小になるようにスケジューリングを行う．後者に関しては，コントロールステップ数の

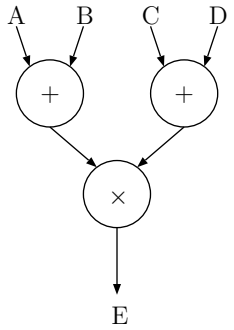

図 2.2: DFG の例

上限が与えられたもとで，資源数が最小となるようにスケジューリングを行う．

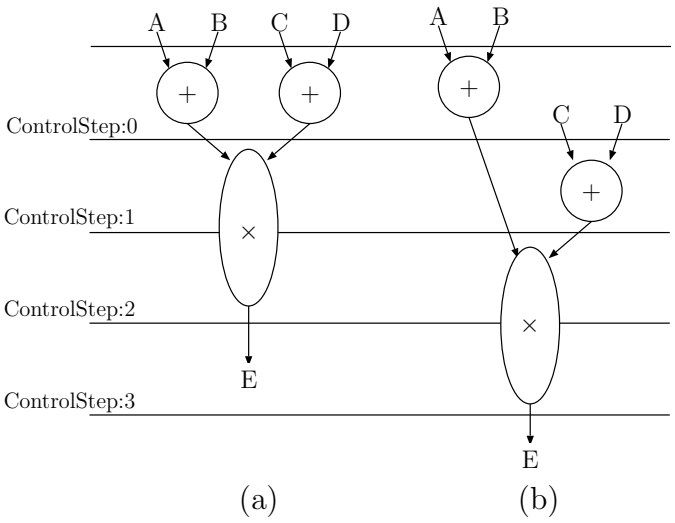

図 2.3: スケジューリング例

2.1.2 レジスタ割当て

レジスタ割当てとは，データパス回路で必要なレジスタ数やデータや演算結果をどのレジスタを利用して保持するのかを決定することである．つまり，演算集合 $\mathcal{O}$ からレジスタ集合 $\mathcal{R}$ への写像 $\xi : \mathcal{O} \mapsto \mathcal{R}$ である．まず，DFG を基に演算結果を保持するための内部変数を割当てる．図 2.4 に例を示す．内部変数はその結果が他の演算で利用される間，値を保持する必要がある．その時間をライフタイムと呼び，内部変数の割当てと同時にライフタイムが決定される．ライフタイムは，コントロールステップのペア (値の保持開始ステップ，値の保持終了ステップ) で表現できる．図 2.5 左にライフタイムの例を示す．次に，内部変数に対してレジスタを割り当てる．1 対 1 に割り当てることも可能ではあるが，一般的にはレジスタ数を最小にするためにレジスタの共有が行われる．内部変数のデータは，ライフタイム間のみレジスタに値を保持できればよいため，ライフタイムが重複しない内部変数は同じレジスタを共有可能である．図 2.5 右に，レジスタ数 4 のレジスタ割当ての例を示す．

2.1.3 演算器割当て

演算器割当てとは，各演算を実現する演算器を決定することである．つまり，演算集合 O から演算器集合 F への写像 $\rho : O \mapsto F$ である．演算器割当てもレジスタ割当て同様，同時に利用しない資源は共有可能であり演算のライフタイムに基づいて割当てが行われ

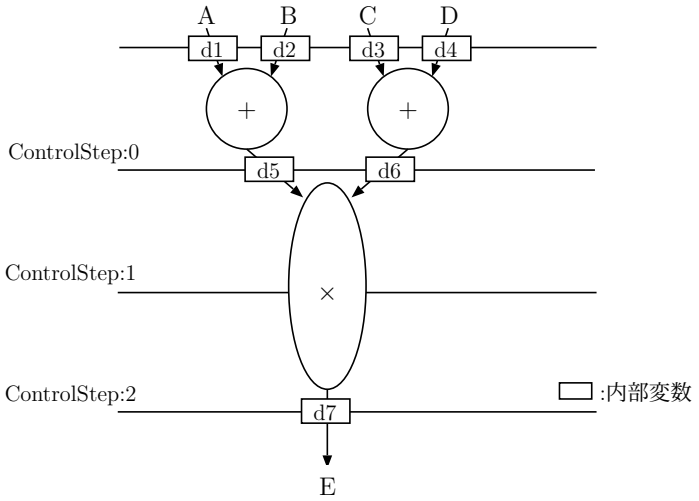

図 2.4: 内部変数割当て例

る．演算のライフタイムとは，コントロールステップのペア (演算の開始ステップ, 演算の終了ステップ) で表現できる．図 2.6 に演算器割当て例を示す．この例では，演算 + は同じコントロールステップに割り当てられている (ライフタイムが重複する) ため，演算器の共有は出来ない．

2.2 セットアップ・ホールド条件

図 2.7 左に示すデータパス回路を例に，タイミングスキュー調整手法の説明を行う．この回路は，演算 o_i の演算結果 (レジスタ r_k に書き込まれる) を入力として，演算 o_j を演算器 f_1 を用いて実行しレジスタ r_l に演算結果が書き込まれる．なお，ここでは演算 o の演算結果をレジスタに書き込む離散的なタイミングをスケジュール $\sigma(o) \in \mathbb{Z}$ とし，クロック周期を t_c とする．レジスタ r_k から演算器 f_1 を通ってレジスタ r_l へ至るまでの最大遅延時間を $d_{\max}^{r_k \to f_1 \to r_l}$, 最小遅延時間を $d_{\min}^{r_k \to f_1 \to r_l}$ とする．演算 $o_i^{(r)}$ は，演算 o_i と同じ出力レジスタを持つ演算であり，演算 o_i の次にそのレジスタに書き込みを行う演算とする．

回路の正常動作は，レジスタが正しいデータをラッチすることである．そのための条件は，セットアップ・ホールド条件と呼ばれる．セットアップ条件とは，演算結果到着後に出力レジスタでの書き込みが行われるための条件である．よって，演算 o_j の演算結果を正しくレジスタ r_l に書き込むためのレジスタ r_k, r_l 間のセットアップ条件は，式 (2.1) の様に書ける．ホールド条件とは，演算結果が書き変わるよりも前に書き込みが行われるための条件である．多くの場合，レジスタは複数のデータによって共有される．図 2.7 においても，レジスタ r_k は演算 $o_i, o_i^{(r)}$ の演算結果を共有している．この場合，演算 o_j の演算結果を正しくレジスタ r_l に書き込むには，演算 $o_i^{(r)}$ の影響がレジスタ r_l に及ぶ時刻より

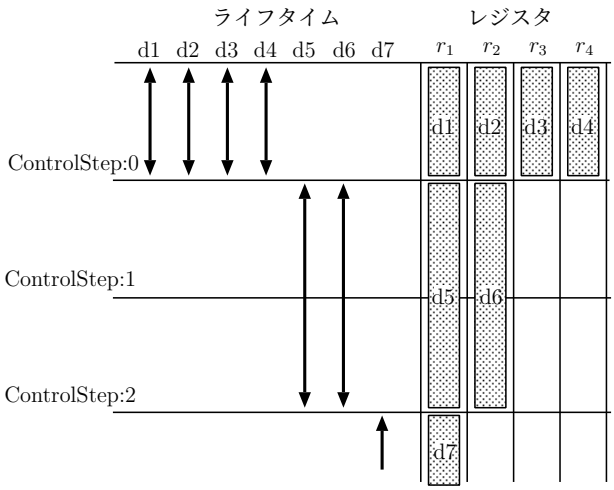

図 2.5: レジスタ割当て例

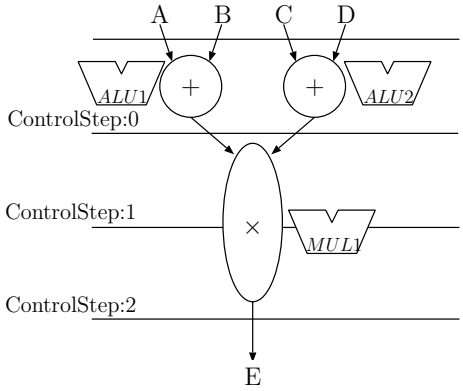

図 2.6: 演算器割当て例

以前に制御信号が到着しなければならない．これがレジスタ r_k, r_l 間のホールド条件であり，式 (2.2) の様に書ける．

$$\sigma(o_i) \cdot t_c + d_{\max}^{r_k \to f_1 \to r_l} \leq \sigma(o_j) \cdot t_c \tag{2.1}$$

$$\sigma(o_j) \cdot t_c < \sigma(o_i^{(r)}) \cdot t_c + d_{\min}^{r_k \to f_1 \to r_l} \tag{2.2}$$

本研究では，それに加えて MUX, レジスタ間のセットアップ・ホールド条件を考慮する．図 2.7 で示した例を用いて，MUX を考慮に入れると図 2.9 のようになる．図中にある，MUX スケジュール $\mu(o_j)$ は，演算 o_j を行うための MUX 切り替えタイミングを表している．MUX から演算器 f_1 を通ってレジスタ r_l へ至るまでの最大遅延を $d_{\max}^{f_1 \to r_l}$ とすると，セットアップ条件は式 (2.3) の様に書ける．ホールド条件は，演算 $o_j^{(f)}$ が演算 o_j と異

なる入力レジスタを持つときのみ必要である．MUX から演算器 f_1 を通ってレジスタ r_l へ至るまでの最小遅延を $d_{\min}^{f_1 \to r_l}$ とすると，ホールド条件は式 (2.4) の様に書ける．

$$\mu(o_j) \cdot t_c + d_{\max}^{f_1 \to r_l} \leq \sigma(o_j) \cdot t_c \tag{2.3}$$

$$\sigma(o_j) \cdot t_c < \mu(o_j^{(f)}) \cdot t_c + d_{\min}^{f_1 \to r_l} \tag{2.4}$$

2.3 タイミングスキュー

図 2.7 右は，信号伝搬遅延のばらつきによりレジスタ間のセットアップ条件に違反しており，タイミングエラーが生じている例である．それに対して，各レジスタ，MUX にタイミングスキューを導入し，全ての演算に対してセットアップ・ホールド条件を満足させる手法が提案されている [6]．図 2.8 にスキュー調整の例を示す．タイミングスキューとは，レジスタ (MUX) の書き込み制御信号 (切り替え制御信号) の到着時刻のズレのことである．ここでは，各レジスタ，MUX に対して独立にスキュー値を設定できると仮定し，レジスタ r_i のスキュー値を $\tau(r_i)$，演算器 f_1 の入力側 MUX のスキュー値を $\tau(f_1)$ とする．なお，MUX には演算器の入力レジスタ切り替えを行うためのものとレジスタの書き込みデータを切り替えるためのものが存在する．ここでは，後者の MUX 切り替え制御信号は，レジスタの書き込み制御信号と同じスキュー値となるとし，前者のスキュー値のみ考慮する．スキューを考慮した場合のセットアップ・ホールド条件は，次の様に書ける．

$$\sigma(o_i) \cdot t_c + \tau(r_k) + d_{\max}^{r_k \to f_1 \to r_l} \leq \sigma(o_j) \cdot t_c + \tau(r_l) \tag{2.5}$$

$$\sigma(o_j) \cdot t_c + \tau(r_l) < \sigma(o_i^{(r)}) \cdot t_c + \tau(r_k) + d_{\min}^{r_k \to f_1 \to r_l} \tag{2.6}$$

$$\mu(o_j) \cdot t_c + \tau(f_1) + d_{\max}^{f_1 \to r_l} \leq \sigma(o_j) \cdot t_c + \tau(r_l) \tag{2.7}$$

$$\mu(o_j^{(f)}) \cdot t_c + \tau(f_1) + d_{\min}^{f_1 \to r_l} > \sigma(o_j) \cdot t_c + \tau(r_l) \tag{2.8}$$

ここで考えるスキュー調整とは，式 (2.5),(2.6),(2.7),(2.8) を満足する様にレジスタスキュー及び MUX スキュー値 τ を計算し回路製造後に設定することである．

2.3.1 スキュー値計算手法

レジスタ r_i のスキュー値 $\tau(r_i)$，演算器 f_j の入力側 MUX のスキュー値 $\tau(f_j)$ は，スキュー制約グラフを用いてグラフの最長路問題として求めることができる．なお，スキュー値 $\tau(r_i), \tau(f_j)$ 以外は全て既知と仮定し説明する．

スキュー制約グラフは，各頂点をスキュー値 $\tau(r_i), \tau(f_j)$ とし，辺にてスキュー値間の制約条件を表した有向グラフである．なお，スキュー制約グラフにおいてレジスタのスキュー値を表す頂点をレジスタスキュー，MUX のスキュー値を表す頂点を MUX スキュー

と呼ぶ．この頂点間の有向辺は，全てのセットアップ・ホールド条件を基に張る．図 2.10 にスキュー制約グラフの例を示す．辺の張り方は式 (2.5),(2.6),(2.3),(2.4) を変形させ，

$$\tau(r_l) \geq \tau(r_k) - (\sigma(o_j) - \sigma(o_i)) \cdot t_c + d_{\max}^{r_k \to f_1 \to r_l} \tag{2.9}$$

$$\tau(r_k) > \tau(r_l) - (\sigma(o_i^{(r)}) - \sigma(o_j)) \cdot t_c - d_{\min}^{r_k \to f_1 \to r_l} \tag{2.10}$$

$$\tau(r_l) \geq \tau(f_1) - (\mu(o_j) - \sigma(o_j)) \cdot t_c + d_{\max}^{f_1 \to r_l} \tag{2.11}$$

$$\tau(f_1) > \tau(r_l) - (\mu(o_j^{(f)}) - \sigma(o_j)) \cdot t_c - d_{\min}^{f_1 \to r_l} \tag{2.12}$$

とし，次のような重みを持つように張る．レジスタ間にセットアップ条件が存在するならば，レジスタ間のセットアップ条件である式 (2.9) の右辺 $\tau(r_k)$ 以降である，$(\sigma(o_i) - \sigma(o_j)) \cdot t_c + d_{\max}^{r_k \to f_1 \to r_l}$ が重みとなる有向辺 $(\tau(r_k), \tau(r_l))$ を張る．この辺をレジスタスキュー間のセットアップ辺と呼ぶ．レジスタ間にホールド条件が存在するならば，レジスタ間のホールド条件である式 (2.10) の右辺 $\tau(r_l)$ 以降である，$-(\sigma(o_i^{(r)}) - \sigma(o_j)) \cdot t_c - d_{\min}^{r_k \to f_1 \to r_l}$ が重みとなる有向辺 $(\tau(r_l), \tau(r_k))$ を張る．この辺をレジスタスキュー間のホールド辺と呼ぶ．MUX，レジスタ間にセットアップ条件が存在するならば，MUX，レジスタ間のセットアップ条件である式 (2.11) の右辺 $\tau(f_1)$ 以降である，$-(\mu(o_j) - \sigma(o_j)) \cdot t_c + d_{\max}^{f_1 \to r_l}$ が重みとなる有向辺 $(\tau(f_1), \tau(r_l))$ を張る．この辺を MUX，レジスタスキュー間のセットアップ辺と呼ぶ．MUX，レジスタ間にホールド条件が存在するならば，MUX，レジスタ間のホールド条件である式 (2.12) の右辺 $\tau(r_l)$ 以降である，$-(\mu(o_j^{(f)}) - \sigma(o_j)) \cdot t_c - d_{\min}^{f_1 \to r_l}$ が重みとなる有向辺 $(\tau(r_l), \tau(f_1))$ を張る．この辺を MUX，レジスタスキュー間のホールド辺と呼ぶ．このように，全ての演算のセットアップ・ホールド条件を基に辺を張り，スキュー制約グラフを作成する．そして，そのグラフに対して最長路問題を解けば，各頂点の最長路が求められ，それが設定すべきスキュー値となる．この時，スキュー値が存在する必要十分条件が次の様に明らかにされている．

定理 1. 全てのセットアップ・ホールド条件を満足するようなスキュー値が存在するための必要十分条件は，スキュー制約グラフにおいて正サイクル *(*サイクルを構成する辺重みの和が正*)* が存在しないことである．

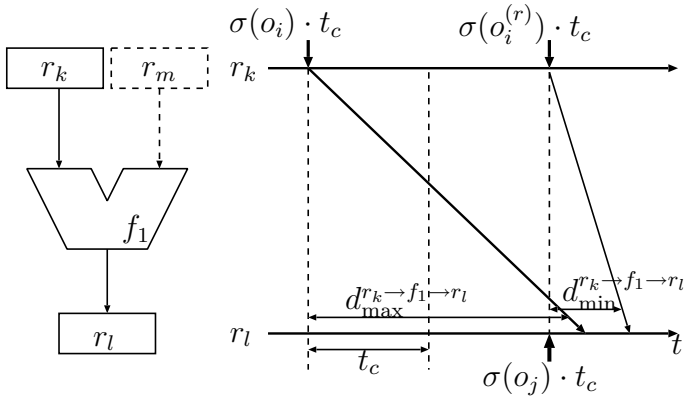

図 2.7: データパス回路とタイミングチャート（タイミング違反例）

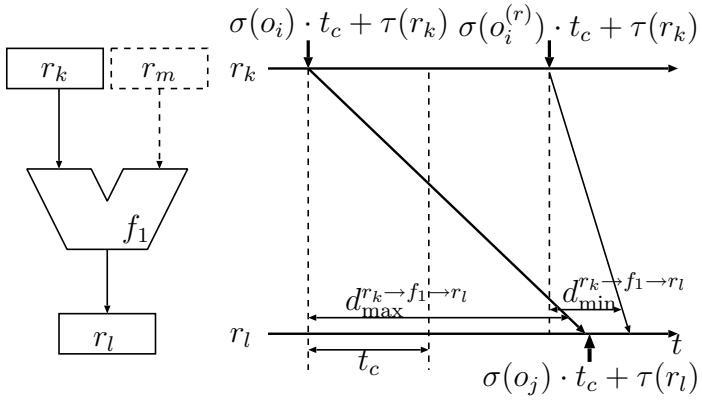

図 2.8: データパス回路とタイミングチャート（スキュー調整例）

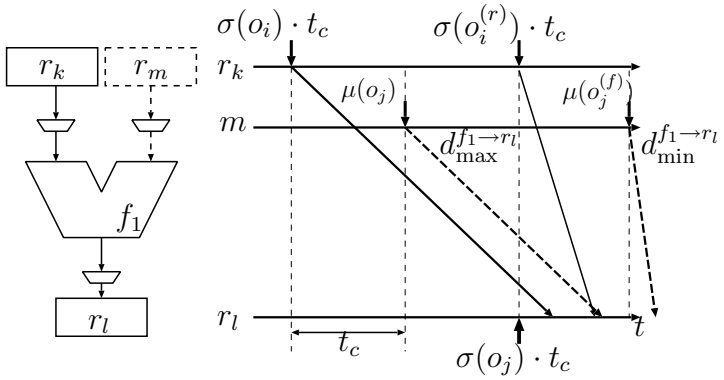

図 2.9: MUX を考慮したデータパス回路とタイミングチャート例

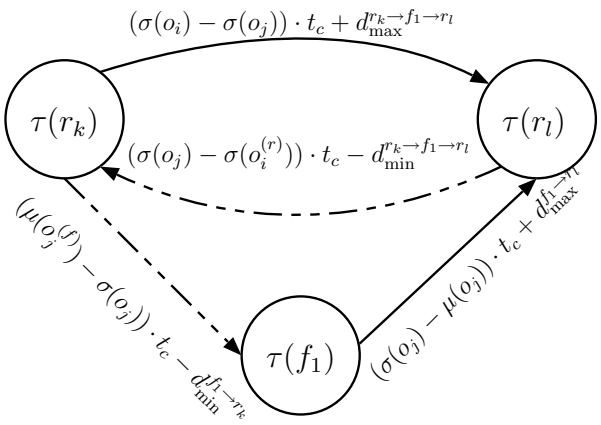

図 2.10: スキュー制約グラフ例

第3章　スキュー調整成功率

本章では，データパス回路の性能評価として導入するタイミングスキュー成功率について述べる．本研究では，データパス回路の性能評価の一つとしてタイミングスキュー調整が成功する確率について考える．LSI 製造時に最大遅延, 最小遅延の値がばらつくことを考慮すると，スキュー制約グラフは，辺重みが確率分布を持つ有向グラフとなる．LSI 製造後のスキュー調整が成功する確率は，このスキュー制約グラフが正サイクルを持たない確率に他ならない．本章では，スキュー制約グラフ及び演算器毎の遅延情報を入力とし，スキュー調整成功率の計算を考える．

3.1　厳密計算法

スキュー成功確率は，スキュー制約グラフの全てのサイクルが正サイクルを持たない確率である．

ステップ 1: スキュー制約グラフ上のサイクルを全列挙.

ステップ 2: 列挙したサイクル $c_1, c_2, \ldots, c_n$ について，各々のサイクル長の確率分布 $L_i (1 \leq i \leq n)$ を計算する．これは，サイクルを構成する各辺の確率分布の和として計算.

ステップ 3: 列挙したサイクルが正サイクルとならない確率 $P(\bigwedge_i L_i \leq 0)$ を計算．この確率は条件付確率を用いて，$P(L_1 \leq 0) \cdot P(L_2 \leq 0 \mid L_1 \leq 0) \cdot \cdots \cdot P(L_n \leq 0 \mid L_1 \leq 0, \cdots, L_{n-1} \leq 0)$ により計算可能である.

により，スキュー調整成功率が計算できる.

確率分布の和の計算は，演算器毎に独立な確率分布を持つとすると次のように計算できる．X, Y を独立な離散型の確率変数とする．その確率分布を $g(x), h(y)$ とする．$X + Y$ の確率分布 $k(z)$ は確率 $P(X + Y = z)$ を考えれば得られ,

$$k(z) = \sum_x g(x)h(z - x) \tag{3.1}$$

となる．g, h が密度関数の時も同様で,

$$k(z) = \int_{-\infty}^{\infty} g(x)h(z - x)dx \tag{3.2}$$

となる．

条件付確率を含む確率の計算は，多次元の確率分布として扱い計算できる．例えば n 個の確率変数 $X_1, X_2, \ldots, X_n$ がそれぞれ $x_i \leq X_i \leq dx_i (1 \leq i \leq n)$ となる確率は，

$$f_{X_1 X_2 \ldots Xn}(x_1, x_2, \ldots, x_n) dx_1 dx_2 \ldots dx_n \tag{3.3}$$

と書ける．X が列挙した各サイクルのサイクル長を表しているとすると，正サイクルとならない確率は積分で，

$$\int_{-\infty}^{0} \int_{-\infty}^{0} \cdots \int_{-\infty}^{0} f(x_1, x_2, \ldots, x_n) dx_1 dx_2 \ldots dx_n \tag{3.4}$$

で求められる．

問題点として，全サイクルの列挙は，多くのインスタンスに於いて現実的でない．また，ステップ 3 の条件付確率を含む確率の計算は，確率分布が確率密度関数の場合，同時確率密度関数の重積分により計算できるが，計算量がスキュー制約グラフ上のサイクル数に対して指数的に増大する．

3.2 ヒューリスティックに基づく計算法

厳密解法での問題に対し，危険度の高いサイクルのみに注目して，近似計算することを考える．

ステップ 1: 遅延量 $d_{\max}, d_{\min}$ の平均値 (又はその他の代表値) を使って，定数重みスキュー制約グラフを作り，予め一度スキュー値計算を行う．そこからサイクル長が大きく，危険度の高いサイクル c_i を複数抽出．

ステップ 2: 抽出された各サイクル c_i について，改めてサイクル長の確率分布 L_i を計算．

ステップ 3: 抽出したサイクルが正サイクルとならない確率 $P(\bigwedge_i L_i \leq 0)$ の計算．

により，近似解を得る．ステップ 1 では，定数重みスキュー制約グラフに少なくとも正サイクルが存在しなければ，全点対間最長路を多項式時間で求めることができ，危険度の高いサイクルを抽出できる．実際に，有向辺 (i, j) を含むサイクル長は，j から i へのパス長 $+(i, j)$ の辺重みとなる．これを各有向辺に対して行い，危険度の高いサイクルの列挙を行う．また，j から i へバックトラックすることにより，サイクルの経路が求められる．ステップ 3 の確率計算では，厳密解法と同様に計算を行う．問題となる重積分の計算は，モンテカルロ法を用いて近似解を得る．

3.3 モンテカルロ法に基づく計算法

近似解を得る他の手法として，与えられたスキュー制約グラフに対し直接モンテカルロ法によるシミュレーションを行う．

ステップ 1: 遅延量 $d_{\max}, d_{\min}$ は正規分布が与えられるとし，その正規分布に従う乱数を生成．

ステップ 2: 与えられたスキュー制約グラフに対して，ステップ 1 で生成した乱数を遅延量として入力．

ステップ 3: 正サイクルが存在するか検証．

ステップ 4: 試行回数分ステップ 1〜3 の繰り返し．

ステップ 5: 正サイクルが存在した回数/試行回数の計算．

により近似解を得る．特定のサイクルを抽出し確率を計算する上記の手法より，精度の高い近似解を得ることができる．

正規分布に従う乱数は，一様乱数からボックス=ミューラー法を用いて生成できる [1]．平均 μ，分散 σ^2 の正規分布 $N(\mu, \sigma^2)$ に従う乱数を生成する例を示す．一様乱数 $(0, 1]$ の要素を x_1, x_2 とし，次のように変換する．

$$z_1 = \sqrt{-2 \cdot \ln x_1} \cdot \sin(2\pi x_2) \tag{3.5}$$

$$z_2 = \sqrt{-2 \cdot \ln x_1} \cdot \cos(2\pi x_2) \tag{3.6}$$

得られた z_1, z_2 が，$N(0, 1)$ に従う乱数となっている．ただし，ln は自然対数である．この乱数に σ をかけ，μ を足すことにより正規分布 $N(\mu, \sigma^2)$ に従う乱数が得られる．

正サイクルの検証は，作成したスキュー制約グラフに対して 2 度最長路長を計算することにより多項式時間で実現できる．1 度目に計算した各頂点の最長路長と，2 度目に計算した各頂点の最長路長が等しければ，正サイクルは存在しない．

確率計算においては有効な計算法であるが，正サイクルとなっているサイクルの特定などが行えない．データパス合成を考える上で，どのサイクルが正サイクルとなっているのか又はなりやすいのかなどの情報が重要となる場合があり，そのような情報を得ることは困難である．

3.4 計算例

スキュー調整成功率の計算例を示すため，モンテカルロ法に基づく計算法を用いて計算を行った．対象とするデータパス回路は，Fast Discrete Cosine Tranform(FDCT)[2] である．アルゴリズム中の演算数は 42 であり，これを ALU 数 3,MUL 数 2 の資源制約の下で

スケジューリングを行って得られた総ステップ数 20 の回路である．各演算器の遅延量として表 3.1 で示す, 最大遅延 $d\max$, 最小遅延 $d\min$ の平均値 E, 分散 V に従う正規分布とした．成功率を比較するため，同一レジスタへの書き戻し箇所が無い (fdct00-0), タイミング制約が厳しい箇所で書き戻しがある (fdct01-1) レジスタ割当てを行い，異なるクロック周期毎に計算を行った．ただし，それぞれスケジューリング，演算器割当ては固定であり，使用レジスタ数は fdct00-0 は 24 個，fdct01-0 は 23 個である．その結果を図 3.1 に示す．

計算結果より，書き戻しがないレジスタ割当てでは，書き戻しが存在するレジスタ割当てよりも最大で 50 ポイント成功率が向上していることが確認できた．ただし，この計算結果はレジスタ間のセットアップ・ホールド条件のみを考慮した計算結果である．

表 3.1: 演算器遅延情報

	$d_{\max}$: $E[ns]$	$d_{\max}$: V	$d_{\min}$: $E[ns]$	$d_{\min}$: V
ALU1	25	4	17	4
ALU2	19	6	10	1
ALU3	36	5	3	6
MUL1	64	12	33	8
MUL2	81	11	40	4

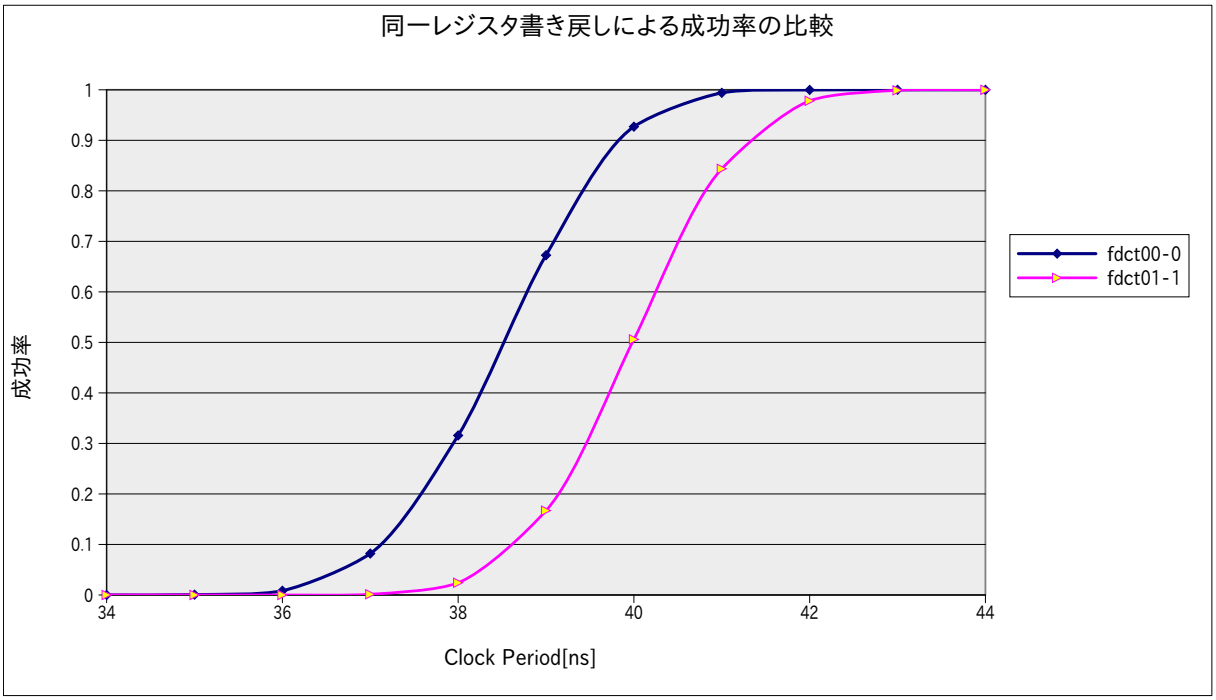

図 3.1: スキュー調整成功率計算例

第4章　スキュー制約グラフにおいて正サイクルが存在しないための設計制約

本研究では，データパス合成を考える第一段階として演算器割当てに着目し，効果的にタイミングスキュー調整が行える割当て問題を考える．まず，タイミングスキュー調整が不可能となる演算器割当て条件を明らかにする．

4.1　データパス合成問題

本章では，各種定義を行い本研究で考える合成問題を明確化する．

演算結果をレジスタに書き込むタイミングであるスケジュール，演算を演算器に割当てる演算器割当て，演算の結果を書き込むレジスタを割当てるレジスタ割当て，入力レジスタを切り替えるタイミングである MUX スケジュールを下記の様に定義する．

演算の集合 $\mathcal{O}$

演算器集合 $\mathcal{F}$

レジスタ集合 $\mathcal{R}$

MUX 集合 $\mathcal{M}$

スケジュール $\sigma : \mathcal{O} \mapsto \mathbb{Z}$

演算器割当て $\rho : \mathcal{O} \mapsto \mathcal{F}$

レジスタ割当て $\xi : \mathcal{O} \mapsto \mathcal{R}$

MUX スケジュール $\mu : \mathcal{O} \mapsto \mathbb{Z}$

スキュー $\tau : \mathcal{R} \cup \mathcal{M} \mapsto \mathbb{R}$

また，$o_j \in \mathcal{O}$ の出力先レジスタを上書きする演算を $o_j^{(r)}$ とし，$o_i \in \mathcal{O}$ を実行する演算器が o_i の次に実行する演算を $o_i^{(f)}$ とする．

本研究では，スキュー調整が高確率で成功するような回路設計を目指している．つまり，スキュー制約グラフ上で正サイクルまたは正サイクルとなる確率が高い危険なサイクルが存在しないような，スケジュール，演算器割当て，レジスタ割当て手法の開発が目標である．

本研究では，データパス合成手法の一つとしてスケジュール後に資源割当てを行うことを想定する．スケジュールの条件は，正サイクルなしの資源割当てが存在し，スケジュール長ができるだけ短くなるように行う．資源割当ての条件は，正サイクルが存在しないように行い，資源数ができるだけ少なくなるように行う．ここでの正サイクルとは，次のようなサイクルだと定義する：簡単化のために遅延量を演算器に独立な正規分布で与えると仮定し，分布を 平均 $+ k \cdot$ 標準偏差 (k : 所望の品質に応じてユーザが設定する定数) で見積もり，サイクル長を計算する．そのときにサイクル長が正となるサイクルを正サイクルと改めて定義する．

4.2 正サイクルが存在するための演算器割当て条件

まず，スケジュールが与えられた上で正サイクルが存在しないような演算器割当てについて考える．

ここでは，演算器割当てが行われた後に正サイクルが存在しないようなレジスタ割当てを行うとすると，正サイクルが存在しないようなレジスタ割当て解の 1 つとして，全ての演算結果を異なるレジスタへの割当てが考えられる．よって，演算に対してユニークにレジスタが割当てられていると仮定し，演算器割当てによって正サイクルが存在するような条件を考える．レジスタ割当てが無制限にされているならば，スキュー制約グラフにおいてレジスタスキューのみからなるサイクルは存在しない．よって，MUX スキューを含むようなサイクルのみ存在する．MUX スキュー間のパス，MUX スキューが 1 つ含まれるようなサイクル，MUX スキューが 2 つ以上含まれるようなサイクルとして場合分けを行い，それらが存在するときの演算器割当ての状況，正サイクルが存在する条件を示す．

4.2.1 MUX スキュー間のパスの条件

スキュー制約グラフにおいて，図 4.1 に示すような MUX スキュー f_1 から f_2 へのパスが存在するための演算器割当ての条件及びパス長を示す． その条件を示すために，MUX スキューについて次の補題が成り立つことを証明する．

補題 4.2.1. *MUX* スキューの入力辺は，レジスタスキューからの辺であり重みはホールド条件のみである．*MUX* スキューの出力辺は，レジスタスキューへの辺であり重みはセットアップ条件のみである．

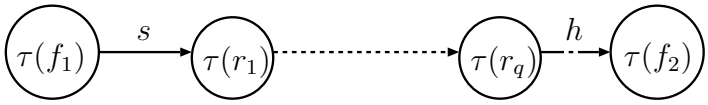

図 4.1: MUX スキュー間のパス

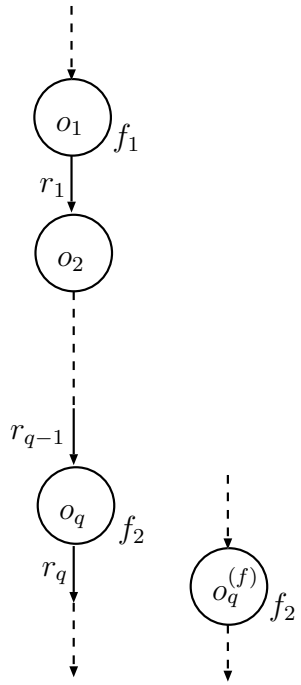

図 4.2: パスが存在する時の演算器割当て

証明. 本研究では，入力側レジスタから演算結果の書き込みレジスタ間，演算器の入力側 MUX から演算結果の書き込みレジスタ間のセットアップ・ホールド条件を基にスキュー制約グラフを作成する．よって，MUX スキュー間の辺，MUX スキューの入力辺であり重みがセットアップ条件，MUX スキューの出力辺であり重みがホールド条件となるような条件は存在しない． □

補題 4.2.1 より MUX スキュー f_1, f_2 間のパスは，少なくとも 1 つのレジスタスキューを含む．また，レジスタ割当て無制限より，図 4.1 中のレジスタスキュー $r_1, r_2, \ldots, r_q$ 間の辺はセットアップ条件のみが重みとなる．以上のようなパスが存在するならば，下記の補題が成り立つ．

補題 4.2.2. *MUX* スキュー f_1 からレジスタスキュー r_1 への辺があるならば，演算器 f_1 を使用し，レジスタ r_1 に値を書き込むような演算が存在する．

証明. 補題 4.2.1 より，MUX スキュー f_1 からレジスタスキュー r_1 への辺はセットアップ条件が重みであることがわかる．この辺は，演算器 f_1 の入力側 MUX から演算結果の書

き込みレジスタ r_1 間に制約条件がある時に張られる．よって，演算器 f_1 を使用しレジスタ r_1 を演算の書き込みレジスタとするような演算が存在する． □

補題 4.2.3. レジスタスキュー $r_1, r_2, \ldots, r_q$ 間にレジスタスキューのみを含むパスがあるならば，依存関係にある演算 $o_1, o_2, \ldots, o_q$ が存在する．

証明. 依存関係にない演算も存在すると仮定する．書き込みレジスタ r_a を持つ演算 o_a 及び演算 o_a と依存関係にない書き込みレジスタ r_b を持つ演算 o_b があるとする $(1 \leq a < b \leq k)$. レジスタ割当てが無制限であるため，演算 o_b の入力レジスタが r_a で無い場合もある．実際に，全ての演算結果に異なるレジスタを割り当てるならば，そのような場合がある．よって，r_a, r_b 間には，入力レジスタ，出力レジスタの関係は存在しない．各演算の入力レジスタ，出力レジスタ間に制約条件が存在するならば，レジスタスキュー間の辺が張られるため，矛盾する． □

補題 4.2.4. レジスタスキュー r_q から *MUX* スキュー f_2 への辺があるならば，演算器 f_2 を使用する演算 o_q が存在し，演算 o_q の次に演算器 f_2 を使用する演算 $o_q^{(f)}$ が存在する．

証明. 補題 4.2.1 より，レジスタスキュー r_q から MUX スキュー f_2 への辺はホールド条件が重みである．このような辺は，演算器 f_2 の入力側 MUX，レジスタ r_q 間にホールド条件が存在する場合に張られる．よって演算器 f_2 を使用する演算 o_q が存在する．

演算 $o_q^{(f)}$ が存在しないと仮定する．演算 $o_q^{(f)}$ が存在しないならば，演算器 f_2 を最後に使用する演算は o_q であり，演算 o_q のホールド条件は必要ない．よって，演算器の入力側 MUX とレジスタ間にホールド条件が存在しないため，レジスタスキューから MUX スキューへの辺が張られることはなく，矛盾する． □

以上より，MUX スキュー f_1 から MUX スキュー f_2 へのパスが存在するならば，図 4.2 のように依存関係にある 2 つの演算 o_1, o_q に演算器 f_1, f_2 が割り当てられ，演算器 f_2 が演算 o_q の次に実行する演算 $o_q^{(f)}$ が存在する．

パス長は，演算 o_1 の MUX，レジスタ間のセットアップ条件：

$$-(\sigma(o_1) - \mu(o_1)) \cdot t_c + d_{\max}^{f_1 \to r}$$

演算 $o_2, o_3, \ldots, o_q$ のレジスタ間のセットアップ条件：

$$\begin{gathered} -(\sigma(o_2) - \sigma(o_1)) \cdot t_c + d_{\max}^{r \to f \to r}, \\ -(\sigma(o_3) - \sigma(o_2)) \cdot t_c + d_{\max}^{r \to f \to r}, \\ \vdots \\ -(\sigma(o_k) - \sigma(o_{k-1})) \cdot t_c + d_{\max}^{r \to f \to r} \end{gathered}$$

演算 o_q の MUX，レジスタ間のホールド条件：

$$-(\mu(o_q^{(f)}) - \sigma(o_q)) \cdot t_c - d_{\min}^{f_2 \to r}$$

の和であり，

$$-(\mu(o_q^{(f)}) - \mu(o_1)) \cdot t_c + d_{\max}^{f_1 \to r} + d_{\max}^{r \to f \to r} + \cdots + d_{\max}^{r \to f \to r} - d_{\min}^{f_2 \to r} \tag{4.1}$$

となる．$d_{\max}^{r \to f \to r} + \cdots + d_{\max}^{r \to f \to r}$ は，依存関係にある演算 $o_2, o_3, \ldots, o_q$ のレジスタ間の最大遅延の和である．

4.2.2 MUX スキューが 1 つ含まれるサイクルの条件

2 頂点からなるサイクルの条件

スキュー制約グラフにおいて，図 4.3 に示すような MUX スキュー f_i ，レジスタスキュー r_j からなるサイクルが存在するための演算器割当ての条件，サイクル長及び正サイクルとなる条件を示す．

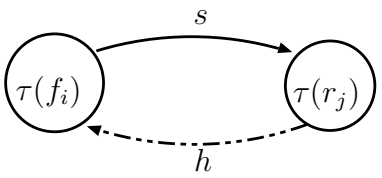

図 4.3: MUX スキューを含む 2 頂点からなるサイクル

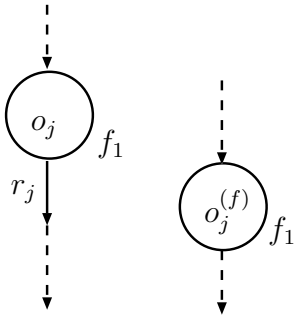

図 4.4: 2 頂点からなるサイクルが存在する時の割当て

補題 4.2.2, 補題 4.2.4 より，MUX スキュー f_i，レジスタスキュー r_j の 2 頂点からなるサイクルが存在するならば，図 4.4 のような演算器 f_i を使用する演算 o_j が存在し，演算 o_j の次に演算器 f_i が使用される演算 $o_j^{(f)}$ が存在する．

サイクル長は，演算 o_j の MUX，レジスタ間のセットアップ条件:

$$-(\sigma(o_j) - \mu(o_j)) \cdot t_c + d_{\max}^{f_i \to r_j}$$

演算 o_j の MUX，レジスタ間のホールド条件:

$$-(\mu(o_j^{(f)}) - \sigma(o_j)) \cdot t_c - d_{\min}^{f_i \to r_j}$$

の和であり，

$$-(\mu(o_j^{(f)}) - \mu(o_j)) \cdot t_c + d_{\max}^{f_i \to r_j} - d_{\min}^{f_i \to r_j} \tag{4.2}$$

となる．

正サイクルとなる条件は，$d_{\max}, d_{\min}$ を 平均 $+ k \cdot$ 標準偏差 で見積もったときのサイクル長が正となる場合である．よって正となる条件は (4.2) 式を変形し，

$$-(\mu(o_j^{(f)}) - \mu(o_j)) \geq \left\lceil \frac{-d_{\max}^{f_i \to r_j} + d_{\min}^{f_i \to r_j}}{t_c} \right\rceil \tag{4.3}$$

となる．

3 頂点以上からなるサイクルの条件

スキュー制約グラフにおいて，図 4.5 に示すような MUX スキュー f_1 ，レジスタスキュー $r_1, r_2, \ldots, r_q$ からなるサイクルが存在するための演算器割当ての条件，サイクル長及び正サイクルとなる条件を示す．

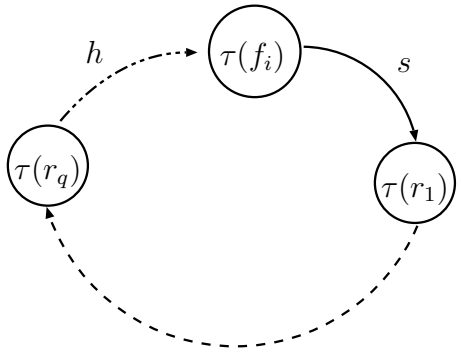

図 4.5: MUX スキューを 1 つ含むような 3 頂点以上からなるサイクル

このようなサイクルは，MUX スキュー f_1 から MUX スキュー f_1 へのパスとみなすことができる．よって 4.2.1 章より，MUX スキューを 1 つ含むような複数の頂点からなるサイクルが存在するならば，図 4.6 のような依存関係にある 2 つの演算 o_1, o_q に同じ演算器 f_1 が割当てられており，演算 o_q の次に演算器 f_1 を使用する演算 $o_q^{(f)}$ が存在する．また，演算 o_1, o_q 間の依存関係にある演算 $o_2, o_3, \ldots, o_{q-1}$ が存在する．

サイクル長は，(4.1) 式のパス長より，

$$\begin{aligned} &-(\mu(o_q^{(f)}) - \mu(o_1)) \cdot t_c \\ &\qquad + d_{\max}^{f_1 \to r} + d_{\max}^{r \to f \to r} + \cdots + d_{\max}^{r \to f \to r} - d_{\min}^{f_1 \to r} \end{aligned} \tag{4.4}$$

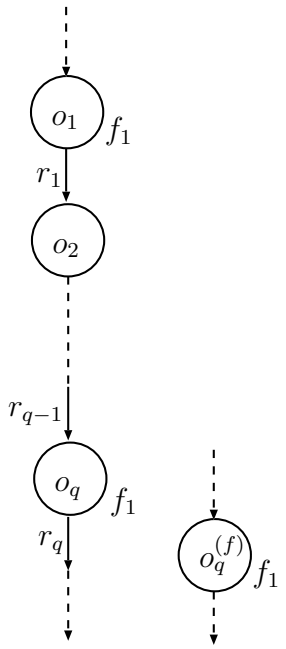

図 4.6: MUX スキューを 1 つ含むような 3 頂点以上からなるサイクルが存在する時の割当て

となる.

正サイクルとなる条件は (4.4) 式より,

$$\begin{aligned} &-(\mu(o_q^{(f)}) - \mu(o_1)) \\ &\quad \geq \left\lceil \frac{-d_{\max}^{f_1 \to r} - d_{\max}^{r \to f \to r} - \cdots - d_{\max}^{r \to f \to r} + d_{\min}^{f_1 \to r}}{t_c} \right\rceil \end{aligned} \tag{4.5}$$

となる.

4.2.3 MUX スキューが 2 つ以上含まれるサイクルの条件

スキュー制約グラフにおいて，図 4.7 に示すような MUX スキュー $f_1, f_2, \ldots, f_p$ からなるサイクルが存在するための演算器割当ての条件，サイクル長及び正サイクルとなる条件を示す.

このようなサイクルは，MUX スキュー f_1 から MUX スキュー f_2 へのパス，MUX スキュー f_2 から MUX スキュー f_3 へのパス，$\ldots$，MUX スキュー f_p から MUX スキュー f_1 へのパスから構成される．よって 4.2.1 章より，サイクルが存在するならば，図 4.8 のように演算器 f_i, f_{i+1} が割り当てられた依存関係にある演算 $o_{1,i}, o_{q,i+1}$ が存在し，演算器 f_{i+1} を演算 $o_{q,i}$ の次に実行する演算 $o_{q,i}^{(f)}$ が存在する．$(i = 1, 2, \ldots, p : if\ i+1 > p\ then\ i+1 := 1)$

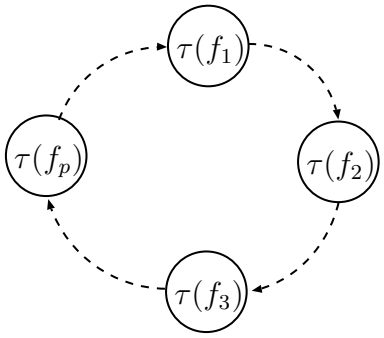

図 4.7: MUX スキューが 2 つ以上含まれるサイクル

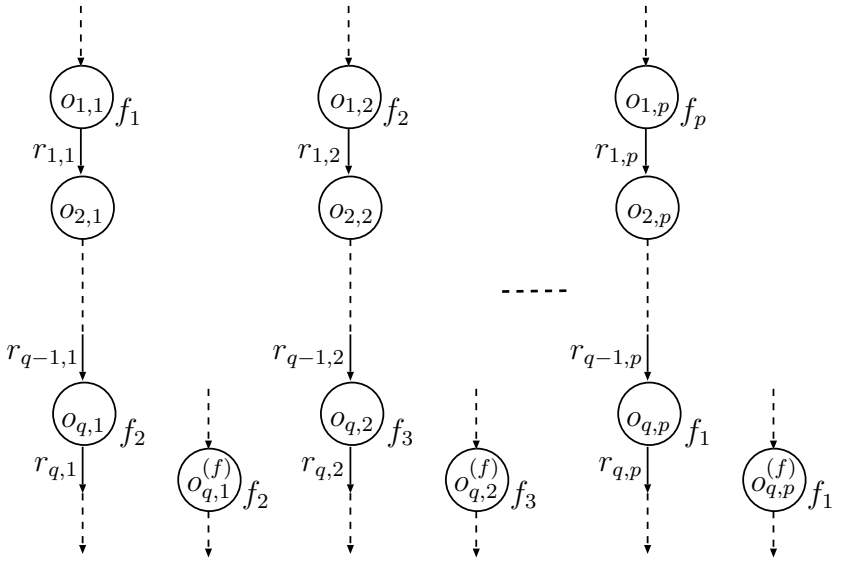

図 4.8: MUX スキューが 2 つ以上含まれるサイクルが存在する時の割当て

サイクル長は各々のパス長の和となるため (4.1) 式より,

$$\sum_{i=i}^{p-1}(-(\mu(o^{(f)}_{q,i}) - \mu(o_{1,i})) \cdot t_c + d^{f_i \to r}_{\max} + d^{r \to f \to r}_{\max} + \cdots + d^{r \to f \to r}_{\max} - d^{f_{i+1} \to r}_{\min})$$
$$-(\mu(o^{(f)}_{q,p}) - \mu(o_{1,p})) \cdot t_c + + d^{f_i \to r}_{\max} + d^{r \to f \to r}_{\max} + \cdots + d^{r \to f \to r}_{\max} - d^{f_1 \to r}_{\min} \tag{4.6}$$

となる.

第5章　スキュー制約グラフにおいて正サイクルが存在しないためのデータパス合成法

本章では，タイミングスキュー調整を考慮した演算器割当て手法を提案する．

5.1　スキュー制約グラフにおいて正サイクルが存在しないための演算器割当て手法

4章で示した条件を基に，スキュー制約グラフにおいて正サイクルが存在しない又は存在しにくい演算器割当て手法を提案する．本研究で考える割当て問題とは，DFG, スケジュールσ ,MUX スケジュールμ, レジスタ間の最大, 最小遅延 $d_{\max}^{r\to f_p\to r}$,$d_{\min}^{r\to f_p\to r}$, MUX，レジスタ間の最大，最小遅延 $d_{\max}^{f_p\to r}$,$d_{\min}^{f_p\to r}$, クロック周期 t_c が入力として与えられ，演算器割当て ρ を得ることである．

本研究における演算器割当て

入力：

DFG

スケジュールσ,MUX スケジュールμ,

レジスタ間の最大, 最小遅延 $d_{\max}^{r\to f_p\to r}$,$d_{\min}^{r\to f_p\to r}$,

MUX，レジスタ間の最大，最小遅延 $d_{\max}^{f_p\to r}$,$d_{\min}^{f_p\to r}$,

クロック周期t_c

条件：

正サイクルが存在しない

目的関数：

演算器数最小化

出力：

演算器割当て ρ

本来ならば，4 章で示すようにサイクルを構成する各辺が正重みを持つ危険度を統計的に評価する必要があるが，ここでは簡単化のためにより定性的に扱う．すなわち，詳細な統計分布を計算する代わりに先ず，辺を余裕のある辺 (重みが正となる確率が十分に小さい辺) と余裕のない辺 (重みが正となる確率が十分に小さいとは言えない辺) に分類する．

MUX-レジスタスキュー間セットアップ辺

個々の設計基準における $d_{\max}^{f\to r}$ を使って

$$(\sigma(o_j) - \mu(o_j)) \cdot t_c \geq d_{\max}^{f\to r} \tag{5.1}$$

となるように設計が行われる．ここで,

$$d_{\max}^{f\to r} + t_c > (\sigma(o_j) - \mu(o_j)) \cdot t_c \geq d_{\max}^{f\to r} \tag{5.2}$$

であるとき余裕のない辺,

$$(\sigma(o_j) - \mu(o_j)) \cdot t_c \geq d_{\max}^{f\to r} + t_c \tag{5.3}$$

であるとき余裕のある辺とする．

MUX-レジスタスキュー間ホールド辺

ここでは，$d_{\min}^{f\to r} < t_c$ と仮定する．これより,

$$(\mu(o_j^{(f)}) - \sigma(o_j)) \cdot t_c > -d_{min}^{f\to r} > -t_c$$

すなわち

$$\mu(o_j^{(f)}) - \sigma(o_j) > -1 \tag{5.4}$$

が設計条件となり,

$$\mu(o_j^{(f)}) - \sigma(o_j) = 0 \tag{5.5}$$

であるとき余裕のない辺,

$$\mu(o_j^{(f)}) - \sigma(o_j) \geq 1 \tag{5.6}$$

であるとき余裕のある辺とする．

レジスタスキュー間セットアップ辺

o_i, o_j が直接的あるいは間接的に依存関係にある演算であって，$\sigma(o_i) < \sigma(o_j)$ であるとする．データフローグラフにおいて o_i の依存関係にある次の演算から o_j までの部分パスを p とすると,

$$\sigma(o_j) - \sigma(o_i) \geq \sum_{x\in p} \text{各演算の } d_{\max}^{r\to f\to r} \text{で決まる最小実行ステップ数} \tag{5.7}$$

となるように設計が行われる．ここで，

$$\sigma(o_j) - \sigma(o_i) = \sum_{x \in p} \text{各演算の } d_{\max}^{r \to f \to r} \text{で決まる最小実行ステップ数} \tag{5.8}$$

であるとき余裕のない辺，

$$\sigma(o_j) - \sigma(o_i) > \sum_{x \in p} \text{各演算の } d_{\max}^{r \to f \to r} \text{で決まる最小実行ステップ数} \tag{5.9}$$

であるとき余裕のある辺とする．この定義の例を図 5.3 を用いて説明する．図 5.3 は，演算 o_1, o_2, o_3, o_4 にそれぞれ演算器 f_1, f_2, f_3, f_4 が割り当てられている．図中の両矢印点線は $\sigma(o_i), \sigma(o_j)$ 間，両矢印破線は部分パス p の区間，両矢印実線は p 中に含まれる演算の $d_{\max}^{r \to f \to r}$ で決まる最小ステップを表している．図 5.3(a) 演算 o_1, o_2 に着目すると，スキュー制約グラフ上の MUX スキュー f_1 から f_2 へのパス中のレジスタスキュー間セットアップ辺はいずれも余裕のない辺となる．図 5.3(b) 演算 o_1, o_4 に着目すると，スキュー制約グラフ上の MUX スキュー f_1 から f_4 へのパス中のレジスタスキュー間のセットアップ辺は少なくとも一つが余裕のある辺となる．

レジスタスキュー間ホールド辺

レジスタ割当ての仮定より，スキュー制約グラフ上にこの辺は形成されないことが言えるため，この辺は考慮しない．

5.1.1 サイクルについて少なくとも 1 辺が余裕のある辺であるための演算器割当て手法

提案手法 1

演算器の共有を行わなければスキュー制約グラフにおいてサイクルが存在しないため，自明な割当て解の一つである．しかしながら，使用する演算器数が増加し回路面積，コストの著しい増大が想像に難くない．有用性のある割当て手法を開発するために，スキュー制約グラフ上のサイクルに少なくとも一つの余裕のある辺が存在すれば，そのサイクルは正サイクルになる確率が十分に小さいと見なすものとし，この制約を満足する割当て手法を提案する．ここでは，

1. 2 頂点からなるサイクル
2. MUX スキューを 1 つ含む 3 頂点以上のサイクル
3. MUX スキューを 2 つ以上含むサイクル

について考慮する必要があり，上記のサイクル 1.，2.，3. を構成する少なくとも 1 辺が余裕のある辺であることを保証するような割当て手法を提案する．1. に関して，通常コントロールステップ数が最小となるようにスケジュールが行われるため，セットアップ辺が余裕のある辺であることは考えず，ホールド辺が余裕のある辺になることを保証する割当て手法を考える．従って，式 (5.6) を満足するような演算を $o_j^{(f)}$ として割当てることにより，少なくとも 1 辺が余裕のある辺であることを保証する．2.，3. に関して 4.2.2，4.2.3 章より，上記の 1. に示した条件を満足するように割当てを行うことにより，サイクルを構成する少なくとも 1 辺が余裕のある重みとなる．なぜならば，サイクルを構成するならば，MUX，レジスタスキュー間のセットアップ辺，ホールド辺が存在することがわかる．また，補題 4.2.4，4.2.2 より，2.，3. のようなサイクルが存在する条件に，1. のようなサイクルが存在する割当て条件も含まれている．よって，上記のように割当てを行うことにより，ホールド辺が余裕のある辺であることを保証できる．具体的な割当て手法について，次に述べる．

式 (5.6) を満足するとは，1 度演算器が使用されると 1 ステップ以上空けなければ次にその演算器は使用できないことを意味する．つまり，

$$\begin{aligned} &\text{演算のライフタイム (演算の開始ステップ, 演算の終了ステップ)} \ = \\ &\quad \text{(演算の開始ステップ, 演算によって決まる最小の終了ステップ} \ + \ 1) \end{aligned} \tag{5.10}$$

として，従来手法のライフタイムに基づく演算器割当てを行い，少なくとも 1 辺が余裕のある辺であることを保証する．ライフタイムに基づく演算器割当てとは，演算器数最小を目標として演算のライフタイムが重ならないように演算を演算器に割当てることである．提案する演算器割当てのアルゴリズムを示す．

ステップ 1: 全ての演算について，ライフタイムを式 (5.10) に従って計算．

ステップ 2: 各演算タイプについて，同じ演算タイプの演算のライフタイムを抽出．

ステップ 3: ライフタイムが重ならないように整列．

ステップ 3 で得られた各列が，割当て演算器に対応している．図 5.1 に例を示す．図左にあるようにライフタイムが計算できたとして，図右のように演算のタイプ毎にライフタイムが重ならないように整列を行う．各演算のタイプの列数が，そのタイプの演算を実行する必要演算器数となる．この例では，$\times$ の列数は 3，$+$ の列数は 2 である．従って，乗算器が 3 つ，加算器が 2 つ必要となる．

5.1.2 サイクルについて少なくともレジスタスキュー間のセットアップ辺 1 辺が余裕のある辺であるための演算器割当て手法

提案手法 2

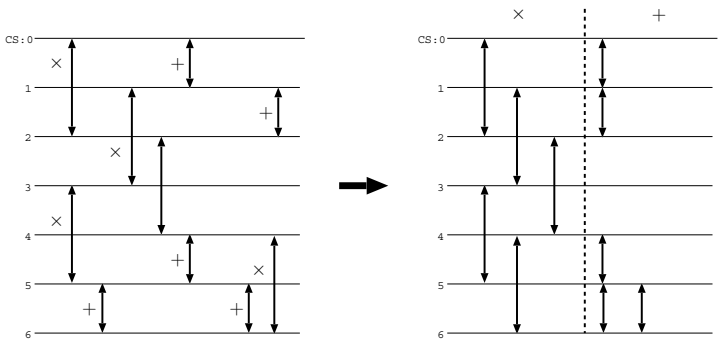

図 5.1: ライフタイムに基づく演算器割当て例

5.1.1 章において提案した割当て手法は全てのサイクルに対して少なくとも 1 辺が余裕のある辺となることを保証する．しかし，2 頂点からなるサイクルを構成する辺がいずれも式 (5.2),(5.5) を満足し余裕のない辺となる辺から構成されていても，正サイクルとならない可能性がある．その時の状況を図 5.2 に示す．この時，最大遅延 $d_{\max}^{f \to r_l}$ がばらついたことを考えると，図中の (a) ばらついたとしてもスキュー調整が行える場合と，図中の (b) ばらつきにより調整不可能な場合が存在する．このことから回路の正常動作を確実に保証することは出来ないが，1 つの余裕のない辺であるセットアップ辺は 1 つのホールド辺によってスキュー調整が可能になると仮定し，制約を更に緩和する．つまり，ここで考える問題は，

1. MUX スキューを 1 つ含む 3 頂点以上のサイクル
2. MUX スキューを 2 つ以上含むサイクル

において，MUX，レジスタスキュー間のセットアップ辺，ホールド辺を除く少なくとも 1 辺が余裕のある辺となるような演算器割当てである．補題 4.2.1 より，

MUX，レジスタスキュー間のセットアップ辺数 $= MUX$，レジスタスキュー間のホールド辺数

であるため，考慮すべき辺はレジスタスキュー間のセットアップ辺のみである．従って，レジスタスキュー間のセットアップ辺の少なくとも 1 辺が余裕のある重みであるような演算器割当てを考える．

資源割当て手法として，度々グラフのカラーリングが用いられる．グラフのカラーリングとは，頂点集合 V, 辺集合 E とする無向グラフ $G = (V, E)$ を入力として，隣接する頂点同士が同じ色にならないように頂点をカラーリングすることである．演算器割当てにおいて，頂点は演算を表し，辺は演算器の共有条件 (辺が存在すれば共有不可) を表す．得られた色が実際に使用される演算器に対応する．本提案手法でも，このグラフカラーリングを用いて解を得ることを考える．よって，

ステップ 1: グラフの作成.

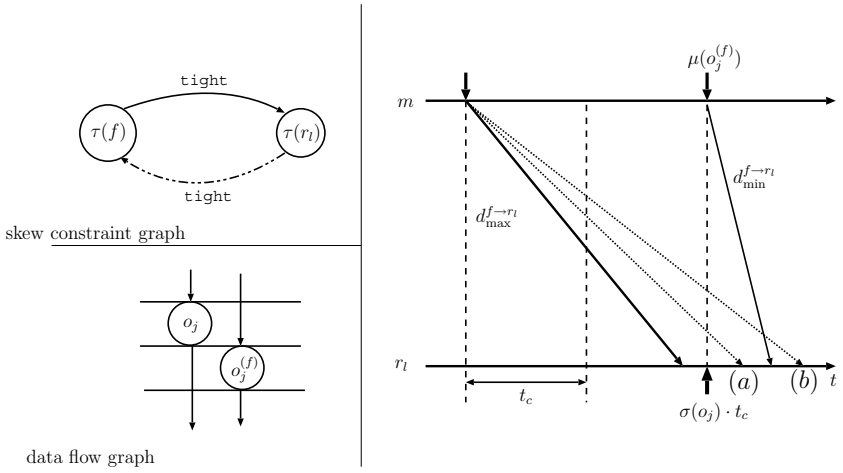

図 5.2: 2 頂点からなるサイクルのいずれの辺も余裕のない辺となっているときの状況

ステップ 2: グラフのカラーリング.

を行い割当て解を得る．ただし，有向辺，無向辺が混在する特殊なグラフを用いたグラフカラーリングとして解を得ることを考える．ここで考えるグラフカラーリングとは，演算を表す頂点集合 V, 演算器の共有条件 (辺が存在すれば共有不可) を表す辺集合 E とするグラフ $G = (V, E)$ を入力として，隣接する頂点同士が同じ色にならないように頂点をカラーリングすることである．ただし辺集合 E は有向辺，無向辺が混在しており，$(x, y) \in E(G), x \in V(G), y \in V(G)$ ならば，色を計算可能な数 π とすると $\pi(x) < \pi(y)$ となるようにカラーリングを行う．$\{x, y\} \in E(G), x \in V(G), y \in V(G)$ ならば，$\pi(x) \neq \pi(y)$ となるようにカラーリングを行う．図 5.4 にグラフカラーリングの例を示す．頂点 v, w 間には無向辺 $\{v, w\}$ が張られており，頂点 w から z への有向辺 (w, z) が張られている例である．頂点 v, w 間には無向辺が存在するので同じ色では塗れない．そこで，頂点 v には 2 を，頂点 w には 1 という色を塗ることにする．次に，頂点 w, z 間には w から z への有向辺が存在するので同じ色が塗れない．なおかつ，有向辺のため w に塗られる色よりも z の方が大きな色を塗らなければならない．先に，頂点 w は 1 を塗るとしたので，2 以上の色を塗らなければならない．ここでは，2 を塗ることが可能であり，2 色でカラーリングが行える．

演算を表す頂点とし，次に示す条件を基に頂点間の辺を張りグラフを作成する．演算 o_x を表す $x \in V(G)$, 演算 o_y を表す $y \in V(G)$ 間 (ただし，$\sigma(o_X) \leq \sigma(o_y)$) に辺を張る条件を示す.

1. 演算 o_x, o_y に依存関係がなく同じ種類の演算かつライフタイムの重なりが無いならば，辺は張らない

2. 演算 o_x, o_y に依存関係がなく同じ種類の演算かつライフタイムの重なりがあるならば，無向辺を張る $(\{x, y\} \cup E(G))$.

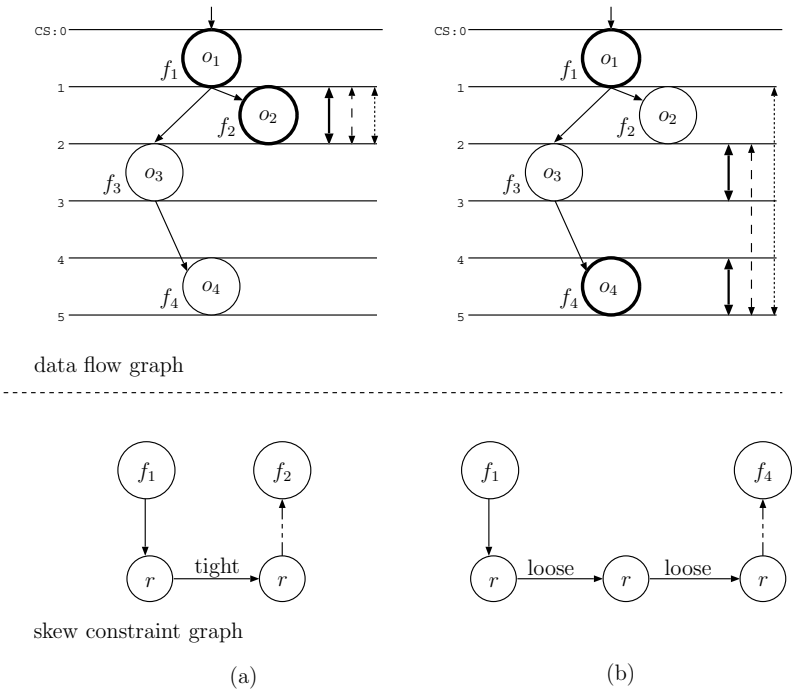

図 5.3: レジスタスキュー間の辺が余裕のない辺，ある辺となる例

3. 演算 o_x, o_y に依存関係がなく異なる種類の演算ならば，無向辺を張る ($\{x, y\} \cup E(G)$).

4. 演算 o_x, o_y に依存関係があり同じ種類の演算かつ式 (5.9) を満足するならば，辺は張らない.

5. 演算 o_x, o_y に依存関係があり同じ種類の演算かつ式 (5.9) を満足しないならば，有向辺を張る ($(x, y) \cup E(G)$)．無向辺を張る ($\{x, y\} \cup E(G)$).

6. 演算 o_x, o_y に依存関係があり異なる種類の演算かつ式 (5.9) を満足するならば，無向辺を張る ($\{x, y\} \cup E(G)$).

7. 演算 o_x, o_y に依存関係があり異なる種類の演算かつ式 (5.9) を満足しないならば，有向辺を張る ($(x, y) \cup E(G)$).

1.，2.，3. に関して，スキュー制約グラフ上のレジスタ間のセットアップ辺は，依存関係にない演算には影響しないため，一般的な演算器の共有条件を基に辺を張る．つまり，同じ種類の演算かつライフタイムが重ならなければ演算器の共有が可能であるとして辺を張らない．一方，異なる種類の演算または同じ種類の演算であってもライフタイムが重なる演算に対しては演算器の共有はできないとして無向辺を張る．4.，5.，6.，7. に関して，スキュー制約グラフ上のレジスタ間のセットアップ辺は，依存関係にある演算とその間にある演算のスケジュールに影響するため，場合分けを行い考える．依存関係にある演算に

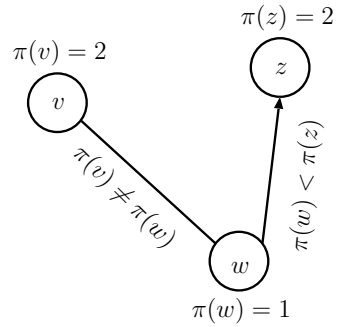

図 5.4: 特殊なグラフカラーリング例

おいて，ライフタイムの重なりは存在しないとして考慮しない．同じ種類の演算である場合，式 (5.9) を満足するようなスケジュールに余裕のある演算ならば，演算器の共有が可能として辺を張らない．一方，同じ演算の種類であっても式 (5.9) を満足しないようなスケジュールに余裕のない演算ならば，スキュー制約グラフにおいて余裕のあるセットアップ辺が存在するサイクルとならない．よって，演算器の共有は行わないとして無向辺を張る．異なる種類の演算である場合，演算器を共有できないため辺を張る．ただし，少なくとも図 4.7 に示すような MUX スキュー f_p から f_1 へのパス中に余裕のあるセットアップ辺が存在することを保証するために，有向辺を導入する．式 (5.9) を満足しないようなスケジュールに余裕のない演算ならば，有向辺を張る．このような演算ならば，スキュー制約グラフにおいて余裕の無いセットアップ辺で構成されるパスとなる．従ってこのパスが，MUX スキュー f_p から f_1 へのパスのように MUX スキュー f_1 へ戻るようなパスとならないことを保証する．そのため，スケジュールに余裕が無い演算ならば有向辺を張り，演算器の順序制約を導入する．

5.1.3 提案手法による演算器割当て具体例

本節では，提案した演算器割当て手法を用いた割当ての具体例を示す．割当て対象とするのは，図 5.5 に示す Differential equation[3] であり，図中に示すようなスケジュールであるとする．なお，演算のライフタイムに基づいた従来の演算器割当て手法における最小演算器数は，乗算器 (MUL) 数 3，加減算器 (ALU) 数 2，シフタ (SHIFT) 数 1 である．

まず，5.1.1 章で提案手法 1 による割当て例を図 5.6 に示す．図中にある両方向矢印の線が各演算のライフタイムを示している．なお，実線部分が演算によって決まる最小のライフタイムを示している．割当て結果より，演算器数は，乗算器数 4，加減算器数 3，シフタ数 1 となった．

次に，5.1.2 章で提案手法 2 による割当て例を図 5.7 に示す．また，演算器と色の対応及び割り当てられた演算を表 5.1.3 に示す．割当て結果より，演算器数は，乗算器 (MUL) 数 4，加減算器 (ALU) 数 2，シフタ (SHIFT) 数 1 となった．

割当て手法の演算器数を比較した結果，従来手法，提案手法 2，提案手法 1 の順に演算器数は増加することを確認した．提案手法 2 は，提案手法 1 と比較して制約を緩和している．提案手法 2 では，依存関係にない 2 つの演算で同じ演算のタイプならば，スケジュールに余裕がなくとも演算器を共有できる．そのため，この具体例では ALU 数に差が生じた．しかし，依存関係にある 2 つの演算で演算のタイプが同じならば，スケジュールに余裕がないと演算器の共有はできず提案手法 1 の制約と同等の制約になる．この具体例では依存関係にある乗算が多くかつスケジュールの余裕はいずれの演算でもほぼないことが見てとれる．そのため，乗算器に差は生じなかった．なお，多くのベンチマーク回路についての検証は今後の課題の 1 つとなっている．

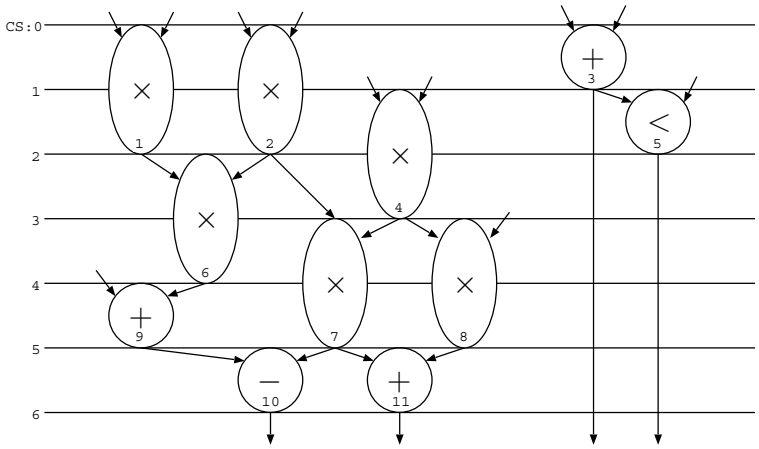

図 5.5: Differential equation の DFG

表 5.1: 演算器と色の対応表

演算器	色	演算
MUL1	$\pi(2)$	4
MUL2	$\pi(4)$	2,7
MUL3	$\pi(6)$	1,8
MUL4	$\pi(8)$	6
ALU1	$\pi(10)$	3,9,11
ALU2	$\pi(11)$	10
SHIFT1	$\pi(5)$	5

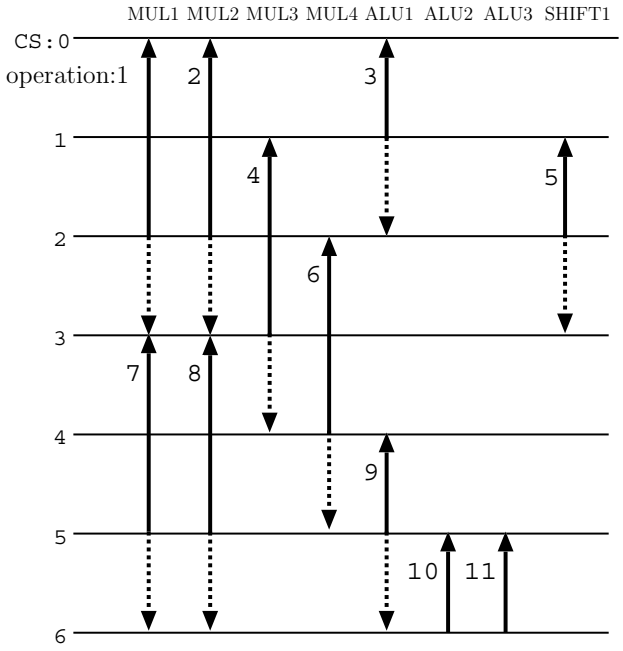

図 5.6: サイクルについて少なくとも 1 辺が余裕のある辺となる割当て結果

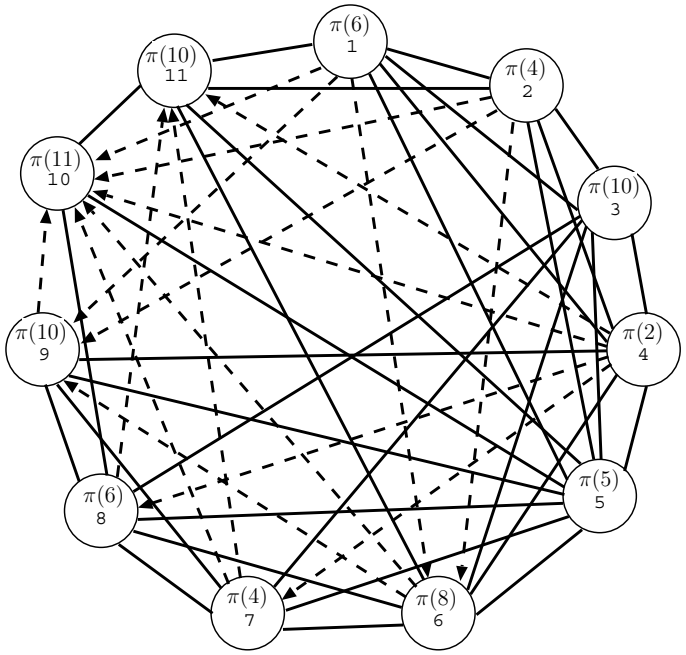

図 5.7: サイクルについて少なくともレジスタスキュー間のセットアップ辺の 1 つが余裕のある辺となる割当て結果

第6章　まとめと今後の課題

本研究では，各レジスタ，MUX の制御信号に意図的にスキューを導入し，回路の正常動作を保証する手法を対象に，その手法が効果的に機能するデータパスの合成理論と手法の開発を目指している．

本稿では，その目標にアプローチする準備としてタイミングスキュー調整成功率を導入すると共に，その計算手法を提案した．次に，タイミングスキュー調整を考慮したデータパス合成を考える第一段階として，演算器割当て問題に取り組んだ．実装すべきアルゴリズムとそれに対する演算実行スケジュールが与えられた上で，まず，タイミングスキュー調整が不可能となる (スキュー制約グラフ上に正サイクルが形成される) 演算器割当て条件を明らかにした．考察の結果，(1) 複数の演算による同一演算器の共有，(2) 依存関係にある 2 つの演算に演算器 f_1と f_2を割当て，同様に f_2と $f_3, \cdots, f_{p-1}$と f_p, f_pと f_1 と演算器を割当てられているとき，スキュー制約グラフ上にサイクルが形成されることがわかった．また割当て対象となった 2 つの演算のスケジュールの差と 2 つの演算の間で実行すべき演算群の計算時間の和の差がタイミング余裕となるが，これが小さいとき，サイクルを構成する辺も余裕のない辺となり正サイクルとなる可能性があることが明らかになった．次いで，こうした考察の結果から，(1) 少なくともサイクルを構成する 1 辺が余裕のある辺となるような演算器割当ての意義を明らかにし，そうした演算器割当て手法を提案した．更に，回路設計の観点から考察し，(2) 制約を緩和した演算器割当て手法の提案を行った．提案した手法について，具体例を用いて割当て解の比較を行った．その結果演算器数は，従来手法，(2) の手法，(1) の手法の順に増加した．なお，他のベンチマーク回路についても同様の結果になるのかは，確認する必要があり今後の課題の 1 つでもある．

今後の課題として，スキュー調整成功率の比較実験が挙げられる．提案手法の有効性を確認するため，種々の回路に対して提案手法による演算器割当てを行い，タイミングスキュー調整にどの程度効果があったのかを確認する必要性がある．また，今回は余裕のある辺とない辺として定性的な評価を行ったが，確率分布に基づいた統計的な設計も考慮したい．更に，タイミングスキュー調整に効果的なレジスタ割当て手法の検討がある．レジスタ割当ては，演算器割当て同様にスキュー制約グラフの形や辺重みを決定付ける要素である．そのため，レジスタ割当てについても正サイクルが存在しないような割当てについて同様に検討する必要がある．タイミングスキュー調整の現実的な問題として，正確な遅延量の測定は困難である．よって遅延量に依存しない調整手法，例えばトライアル&エラーによるタイミングスキュー調整などが考えられ，興味深い検討課題の 1 つである．

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，博士課程 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] G. E. P. Box and M. E. Muller. A note on the generation of random normal deviates. *Ann. Math. Stat.*, Vol. 29, pp. 610–611, 1958.

[2] Keisuke Inoue, Mineo Kaneko, and Tuyoshi Iwagaki. Optimal register assignment with minimum-path delay compensation for variation-aware datapaths. *IEICE Trans.Fundamentals*, Vol. E92-A, No. 4, pp. 1096–1105, 2009.

[3] Tien-Chien Lee, Niraj K. Jha, and Wayne H. Wolf. Behavioral synthesis of highly testable data paths under the non-scan and partial scan environments. *30th ACM/IEEE Design Automation Conference*, pp. 292–297, 1993.

[4] Jaskirat Singh and Sachin Sapatnekar. Statistical timing analysis with correlated non-gaussian parameters using independent component analysis. *In Proc.ACM/IEEE DAC*, pp. 155–160, 2006.

[5] Akihiro Takamura, Masashi Kuwako, Masashi Imai, Taro Fujii, Motokazu Ozawa, Izumi Fukasaku, Yoichiro Ueno, and Takashi Nanya. Titac-2:an asynchronous 32-bit microprocessor based on scalable-delay-insensitive model. *Proceedings of ICCD'97*, pp. 288–294, 1997.

[6] Obata Takayuki and Mineo Kaneko. Control signal skew scheduling for rt level datapaths. *IEICE Technical Report*, Vol. 104, No. 738, pp. 13–17, 2005.

[7] Paul S. Zuchowski, Peter A. Habitz, Jerry D. Hayes, and Jeffery H. Oppold. Process and environmental variation impacts on asic timing. *Proc. International Conference on Computer Design*, pp. 336–342, 2004.

[8] 橋爪裕子, 大谷直毅, 高島康裕, 中村祐一. 離散遅延値を持つ pde を用いたクロックデスキュー手法. 情報処理学会研究報告 SLDM-130, Vol. 39, pp. 45–50, 2007.

[9] 岡田健一, 藤田智弘, 小野寺秀俊. トランジスタ製造ばらつきにおけるチップ内特性変動を考慮した統計的遅延解析手法. 信学技報, Vol. 101, pp. 7–12, 2001.

[10] 柳大吾, 築山修治. 統計的静的遅延解析手法における相関の取り扱いについて. 信学技報, Vol. 149, pp. 19–24, 2002.

[11] 東京大学教養学部統計学教室（編）. 統計学入門. 東京大学出版会, 1991.