修 士 論 文

タイミングスキュー調整特性に優れた高性能LSIのための高位合成手法

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

春田　洋佑

2012年3月

修 士 論 文

タイミングスキュー調整特性に優れた高性能LSIのための高位合成手法

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　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

0910050 春田　洋佑

提出年月: 2012 年 2 月

概 要

集積回路の益々の微細化と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増大している。これまではタイミングマージンを十分確保することでこれに対処してきたが、過剰なタイミングマージンは回路の速度性能の低下を招く。近年、この問題に対して、チップ製造後の回路チューニングの手法が提案されている。本稿では特に製造後のタイミングスキュー調整に注目し、製造ばらつきに対して、高性能な集積回路を高い歩留まりで製造するための回路設計手法を検討、提案する。

製造後タイミングスキュー調整では、遅延量が調整できる素子（PDE）をクロック分配経路にあらかじめ挿入しておき、製造後のチップ個別の信号遅延量に応じて、タイミングスキュー（クロック源から FF までの個別のクロック信号遅れ時間）を調整することで、回路を正常動作させる。

ここで、LSI 製造時に生じる回路中の遅延ばらつきに対して、スキューを調整することによって回路を正常動作させることができる確率をスキュー調整成功確率と呼ぶこととする。また、回路の構造（信号伝搬経路）と動作のタイミング (信号伝搬のタイミング）に大きな影響を与える高位合成の中での最適設計（スキュー調整成功確率最大化）を考える。

高位合成の中の資源割り当ては、実装すべき計算アルゴリズムとその演算スケジュールを入力とし、演算やデータの生存期間が競合しない条件のもとで、演算やデータを限られた演算器やレジスタに割り当てる問題であり、レフトエッジと呼ばれる代表的な手法がある。レフトエッジや、その後タイミングスキューを考慮して提案された minTc、color といった資源割り当て手法ではスキュー調整成功確率を直接考慮していないため、所望の最適解を得られるとは限らない。こうした手法に対して本研究では、スキュー調整成功確率を直接評価しながら資源割り当てを行う手法を検討する。原理的にはすべての資源割り当て解を列挙して、それぞれに対してスキュー調整成功確率を評価し、その中の最善の解を求めることにより最適解が得られる。しかし、すべての資源割り当て解を列挙して評価することは解空間が膨大で、計算時間的に現実的ではない。

第一の提案手法 (マージ法) では、まずデータと演算をすべて異なるレジスタと演算器に割り当てた初期解を用意する。その後、生存期間競合の関係から共有可能なデータあるいは演算の２つの組み合わせを列挙し、それぞれについて実際に共有を行った際のスキュー調整成功確率を評価する。スキュー調整成功確率が最善の組み合わせを１つ選択し、資源割り当てを更新する。共有候補がなくなる、あるいは所望の資源量に達するまでこの操作を繰り返し、資源を削減する。このマージ法はその時々の最善の資源共有を繰り返すグリーディー手法であり、資源数制約を保証できない、繰り返し共有の後半で評価が急激に悪化するなどの問題がある。

第二の手法として資源数制約を満たしながらスキュー調整成功確率を考慮する資源割り当て（並列レフトエッジ法）を提案する。

通常のレフトエッジ法が常に1つの資源への割り当てを考えるのに対して、並列レフトエッジ法では指定された資源のすべてに対して、同時並行的に割り当てを行う。その手続きはスケジュールの第１実行ステップから順に、各実行ステップｓ毎に、ｓからライフタイムが始まる演算、データをｓにおいてアイドルな演算器、レジスタに割り当てを行うものである。１つの実行ステップｓでの割り当ての決定にあたっては、まず候補の演算(データ)の1つ1つに対して、アイドルな演算器(レジスタ)の１つ１つに割り当てした際のスキュー調整成功確率を評価する。次いで、候補演算(データ)集合Xsとアイドル演算器(レジスタ)の集合Ysを部集合とする完全2部グラフを構成し、演算ー演算器(データーレジスタ)辺に、対応する割り当てを行った際のスキュー調整成功確率を辺重みとして付加する。そうした上で、最小辺重み最大のXsからYsへのマッチングMsを計算し、Msに含まれる辺に対応する演算の演算器への(データのレジスタへの)割り当てを採用する。

既存手法と提案手法について、回路合成実験を行ったところ、並列レフトエッジ法は通常のレフトエッジ法に対して常に優れた解を得るものの、minTc、color手法に対しては、幾つかの例外を除いて、その解は劣るものであった。これは、並列レフトエッジ法の大域的最適解計算能力の低さが主因と考えられる。その一方で、並列レフトエッジ法がminTc、colorよりもすぐれた解を得ている例もあり、スキュー調整成功確率を正しく評価しながら設計を行うことの重要性が示された。

今後、スキュー調整成功確率評価の正しさを継承する、より優れた解探索手法の開発が望まれる。また、タイミングスキュー調整を考慮したスケジューリングや演算器の遅延の分布を把握するための統計的遅延解析の採用も今後の課題である。

目 次

第1章　はじめに

近年の集積回路の微細化、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増大している [1]。こうした遅延ばらつきによる回路の製造歩留まり低下が問題となっている。遅延ばらつきの原因としては製造条件の揺らぎや電圧変化、温度変化などさまざまな原因が挙げられる。

遅延ばらつきによる回路の製造歩留まり低下の解決策として、これまではタイミングマージンを十分確保することで対応してきた [2]。しかし、過剰なマージンを確保することは回路性能の低下を招き、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がある。

タイミングマージンを確保するチップ製造前の設計とは別に、チップ製造後の回路チューニングの手法が提案されている。実際の方法として、遅延量が調整できる素子 (Programmable Delay Element,PDE) をクロック分配経路に挿入して、制御信号が各レジスタに到着するタイミングを積極的にずらすことにより、回路をより高速に動作させる製造後スキュー調整がある [3][4][5]。この手法はチップ個別の性能を十分に引き出した高性能集積回路の実現を可能にするが、製品としての歩留まりやスキュー調整によって達成されうる性能は、回路設計に依存している。

チップ製造前の設計とチップ製造後の回路チューニングによる回路の正常動作する確率を図 1.1 に示す。横軸はクロック周期であり、縦軸は回路の正常動作する確率を表している。1.1 よりチップ製造後の回路チューニングがチップ製造前の設計より歩留まりが向上することがわかる。

本研究では製造ばらつきに対して、この製造後スキュー調整にて高性能な集積回路を高い歩留まりで製造するための設計手法を検討、提案する。回路を正常動作させるために、データの取り込みを確実に行うタイミング制約であるセットアップ条件とホールド条件をすべての演算について書きだし、それらすべてを同時に満足するような各レジスタと演算器のタイミングスキュー値（クロック信号からの個別の遅れ時間）を決定する必要がある。タイミングスキュー値の決定は、１つ１つのタイミング制約を有向辺とするスキュー制約グラフ上で特別に定めた始点からすべての頂点への最長パス長を求めることによってなされる。このことから、遅延ばらつきのもとで回路を正常動作させる確率（スキュー調整成功確率）の最大化は、スキュー制約グラフ上にある全サイクルの辺重みの総和が負になる確率の最大化に等しい。また、高位合成は動作記述からレジスタ転送レベル回路記述に変換する工程であり、回路中の信号の流れとそのタイミングを決める重要な設計段階である。

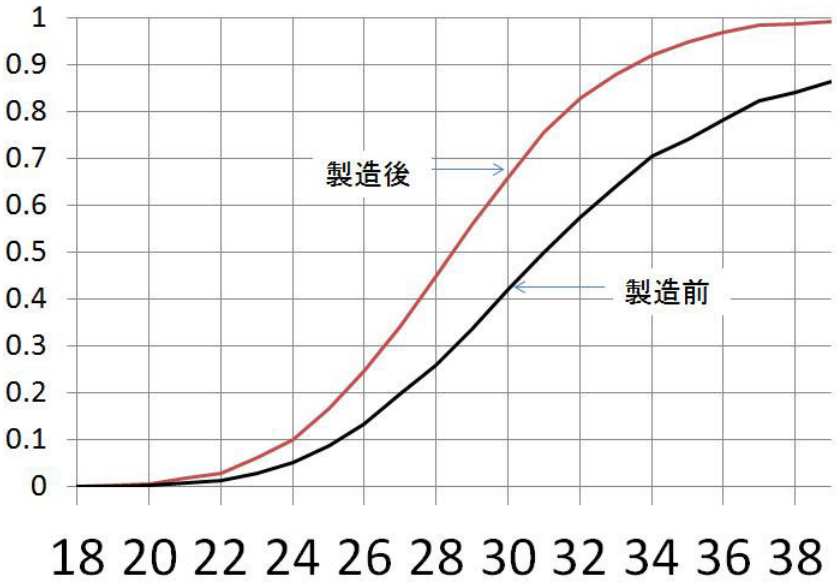


図 1.1: 正常動作する確率の比較

本研究では各レジスタと演算器にタイミングスキューを導入することでスキュー制約グラフを生成し、スキュー調整成功確率を計算し、確率の値をもとに資源割り当てする手法を提案した（merge 法)。また、スキュー調整成功確率を考慮しながら資源数制約を満たす割り当て手法も提案した（並列レフトエッジ)。

本稿は以下のように構成する。第 2 章では、回路が正常動作するタイミング制約を定義し、タイミングスキューを導入することで制約を満たす方法を説明する。第 3 章では、スキュー調整成功確率について説明し、議論する問題を設定し関連研究について述べる。第 4 章では、スキュー調整成功確率の計算方法について説明する。第 5 章と第 6 章では、割り当て方法と実験方法を説明し、実験結果の評価を行う。第 7 章でまとめと今後の課題を述べる。

第2章　データパス回路のタイミング制約

2.1　セットアップ条件とホールド条件

図 2.1 に示すデータパス回路を例に，本論文にて対象とするタイミングスキュー調整の概要を説明する．本論文では演算を o，演算結果を d，演算器を f，レジスタを r で表す．

図 2.1(a) のスケジュール済みデータフローグラフに対して，図 2.1(b) のデータパス回路を使用し，d_1 と d_3 を r_1 に，d_2 を r_2 に割当する．この状況において，o_2 が演算器 f_j で実行される際の動作タイミングを図 2.1(c) に示す．σ_s,σ_e $\in\mathbb{Z}_+$ は演算の開始ステップと終了ステップを表し，D_{max} と D_{min} は r_1 から f_j を通り r_2 に至る最大遅延と最小遅延，T_c はクロック周期である．

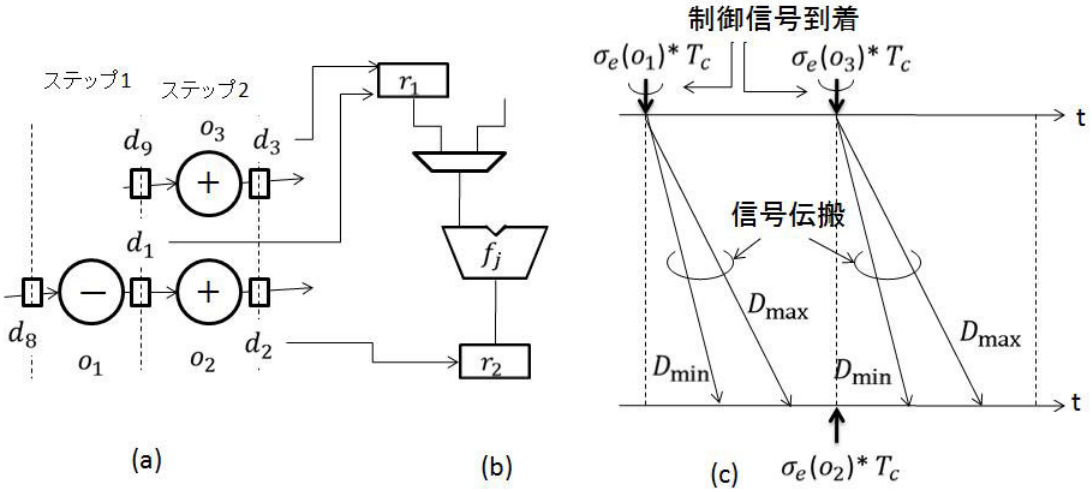


図 2.1: タイミング制約

データの取り込みをタイミング的に正しく行うためには，演算結果が書き込み先のレジスタに到着した後に，書き込み制御信号が到着することを規定するセットアップ条件 (式 (2.1)) と，レジスタへの書き込み制御信号が到着する前に，演算結果が失われないことを規定するホールド条件 (式 (2.2)) を満たすことが必要かつ十分である．

演算 o_2 についてのセットアップ条件：

$$\sigma_e(o_1) \times T_c + D_{max} \leq \sigma_e(o_2) \times T_c \tag{2.1}$$

演算 o_2 についてのホールド条件：

$$\sigma_e(o_3) \times T_c + D_{min} > \sigma_e(o_2) \times T_c \tag{2.2}$$

f_j の入力側マルチプレクサと出力レジスタ間にも，同様のセットアップ条件とホールド条件が存在する．

製造後の LSI 回路には，クロック信号の各 FF への到着時刻のばらつきやレジスタ間やマルチプレクサとレジスタの間の信号遅延時間のばらつきが存在する．クロックの到着時刻のばらつきを Δ_{ri}，信号伝搬遅延のばらつきを D_{max}，D_{min} が含むものとすると，チップ製造後のタイミング制約式は式 (2.3)(2.4) となる．

$$\sigma_e(o_1) \times T_c + \Delta_{r1} + D_{max} \leq \sigma_e(o_2) \times T_c + \Delta_{r2} \tag{2.3}$$

$$\sigma_e(o_3) \times T_c + \Delta_{r1} + D_{min} > \sigma_e(o_2) \times T_c + \Delta_{r2} \tag{2.4}$$

このとき，Δ_{ri}，D_{max}，D_{min} の設計値からのずれにより不等式を満たさず，回路が正常動作しなくなる可能性が発生する．

2.2 タイミングスキュー調整

タイミングエラーに対して，各レジスタと演算器の入力マルチプレクサに制御タイミングスキューを導入することにより，セットアップ条件とホールド条件を満足させることを考える．タイミングスキュー導入方法として，チップ製造後に遅延量が調整できる素子 (Programmable Delay Element,PDE) をクロック分配経路に挿入する (図 2.2 (a))．図 2.2(b) はスキューを導入した後において演算 o_2 が演算器 f_j で実行される際の動作タイミングを表している．なお，レジスタ r_1 と r_2 のスキュー値を τ_{r1} と τ_{r2} としている．また，このときのレジスタ間のセットアップ条件とホールド条件は式 (2.5)(2.6) となる．

$$\sigma_e(o_1) \times T_c + \Delta_{r1} + \tau_{r1} + D_{max} \leq \sigma_e(o_2) \times T_c + \Delta_{r2} + \tau_{r2} \tag{2.5}$$

$$\sigma_e(o_3) \times T_c + \Delta_{r1} + \tau_{r1} + D_{min} > \sigma_e(o_2) \times T_c + \Delta_{r2} + \tau_{r2} \tag{2.6}$$

ここで $r_1(r_2)$ のスキュー $\tau_{r1}(\tau_{r2})$ の調整にて $\Delta_{r1}(\Delta_{r2})$ を打ち消せるものと考え，以下では $\Delta_{r1}(\Delta_{r2})$ を省いて議論する．

2.3 スキュー値計算手法

各レジスタ，マルチプレクサのスキュー値は，すべての演算についてのセットアップ条件とホールド条件を同時に満足する連立不等式の解として定められる．この具体的解法の

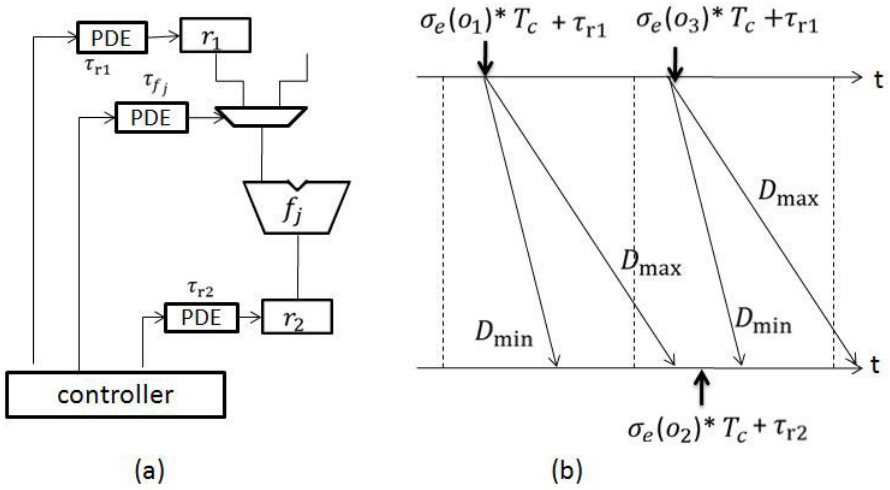


図 2.2: スキュー導入によるエラー回避

一つとして，スキュー制約グラフを利用する手法がある．スキュー制約グラフは，各レジスタや演算器のスキュー値 $(\tau_{r1},\tau_{r2},\cdots)$ を頂点として，一つ一つのタイミング制約を有向辺とする有向グラフである．例えば，式 (2.5) は式 (2.7) と変形でき，これに対応して重み $(\sigma_e(o_1) - \sigma_e(o_2)) \times T_c + D_{max}$ の有向辺（τ_{r1}, τ_{r2}）を設ける．

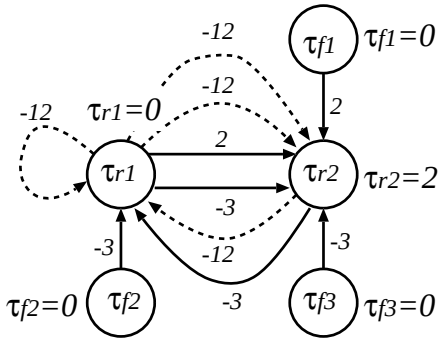
$$\tau_{r2} \geq \tau_{r1} + (\sigma_e(o_1) - \sigma_e(o_2)) \times T_c + D_{max} \tag{2.7}$$

また設定するスキュー値を 0 以上の値に限定するために，特別な頂点 s と s から他のスキュー値頂点への重み 0 の辺を加える．遅延量 D_{max}, D_{min} を定数値として構成されたスキュー制約グラフ G が正サイクルを持たないとき，G 上で s から各頂点への最長パス長を求め，それをその頂点のスキュー値とすることですべての制約を満たすスキュー値が得られる．

ここでスキュー制約グラフの例を示す．図 2.2(a) のレジスタ，演算器に図 2.1(a) のデータ，演算を割り当てる．d_8 と d_9 と d_2 を r_1 に，d_1 と d_3 を r_2 に割り当て，演算はすべて個別の演算器に割り当てた構成に対するスキュー制約グラフが図 2.3 である．スキュー制約グラフ上の実線はセットアップ条件の辺，破線はホールド条件の辺を意味する．なお，加算器の遅延量は最大遅延 17[ns], 最小遅延 12[ns] とし，減算器の遅延量は最大遅延 22[ns], 最小遅延 12[ns] としている．クロック周期は 20[ns] とする．このとき，τ_{r2} を 2[ns]，ほかのスキュー値を 0[ns] とするスキュー調整が求められる．

なお，制約を満足するスキュー値の存在性に関して，次の定理が成り立つ．

定理 2.3.1 すべてのセットアップ条件とホールド条件を満足する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正サイクルが存在しないことである．



**Vertex s and its outgoing edges are omitted*

図 2.3: スキュー制約グラフ：実線はセットアップ条件の辺重み，破線はホールド条件の辺重みを表す

第3章　遅延ばらつきとスキュー調整

3.1　スキュー調整成功確率

前節の議論から，チップ毎のばらつきのインスタンスに対して，スキュー調整の成否（スキュー制約グラフ中の正サイクルの無有）が決まるが，一方，ばらつきを持ったチップの集合に対しては，スキュー調整が成功する割合（スキュー調整成功確率）を考えることができ，これが製造後のスキュー調整によって達成される製造歩留まりとなる．

回路中のレジスタ間あるいはマルチプレクサーレジスタ間の信号伝搬遅延が，チップ毎にばらつく状況は，スキュー制約グラフにおいて辺重みが確率分布を持つことと捉えることができる．こうした見方に立つと，スキュー調整成功確率は辺重みが確率分布を持つ確率的スキュー制約グラフ（構造は資源割り当てにて決まる）が正サイクルを持たない確率として計算される．

3.2　問題設定

トップダウンによるLSIの設計フローはまず仕様を決めるアルゴリズム設計、つぎに、ハードウェアとして実現するため動作記述をレジスタ転送レベル（RTL）記述回路に変換する高位合成、そして、RTL記述を論理ゲート回路に変換する論理合成、最後にマスクパターンを設計する物理レベル合成という流れである。今回はLSIの設計フローの高位合成に注目する。

高位合成の流れを説明する。まず、与えられた動作記述をコントロール／データフローグラフ（CDFG）とよばれるグラフで表現する。コントロールフローは動作記述の制御分の構造を示す。また、データフローは演算操作の構造を示す。次に、CDFGに現れる各演算をどのステップに実行するかのスケジュールを決める。各ステップに演算を割り当てることをスケジューリングという。スケジューリングにおいては、与えられた制約のもとで最適化の目標を決め、スケジュールを決める。制約や最適化の目標としては時間や面積がある。資源割り当ては、スケジューリングされたCDFGに現れる各演算や変数に、具体的な演算器やレジスタを割り当てる処理をいう。最後に、割り当てられた演算器やレジスタ間の結線を実現する。これによりデータパスが生成される。生成されたデータパスで、CDFGの動作を実現するために、制御信号を生成するコントローラを生成する。コントローラとデータパスが生成されると、レジスタ転送レベル回路記述が生成されたことにな

る。本研究では高位合成の資源割り当てに注目し、簡単化のためコントロールフローは対象から外して、データフローグラフを対象とする。

スケジュール済みデータフローグラフ，演算器情報，クロック周期，資源数の上限を入力とし，スキュー調整成功確率を最大化する資源割り当てを出力する問題を考える．

図 3.1 のスケジュール済みデータフローグラフを例に資源割り当てとスキュー調整成功確率の関係性を示す．なお，加算器の最大遅延を平均 20[ns] 標準偏差 4[ns]，最小遅延を平均 12[ns] 標準偏差 2.4[ns] とする正規分布とし，クロック周期を 24[ns] とする．また，レジスタ数は下限の 2 とする．

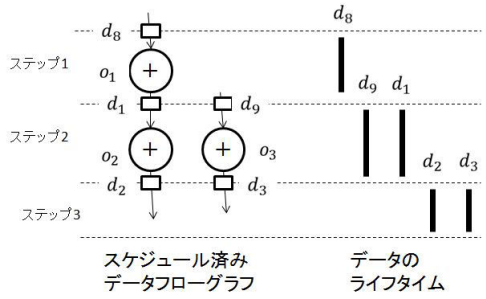


図 3.1: 製造後スキュー調整を考慮した資源割当

資源割り当てを行い，スキュー制約グラフを作成し, スキュー調整成功確率を評価したのが図 3.2 である．r_1 に d_8 と d_9 と d_2，r_2 に d_1 と d_3 を割り当てた第一の割り当て解に対するスキュー調整成功確率は 0.85 となる．一方，r_1 に d_8 と d_9 と d_3，r_2 に d_1 と d_2 を割り当てた第二の割り当て解に対してはスキュー調整成功確率は 0.71 である．このように資源割当がスキュー調整成功確率に大きく影響を与えることがわかる．

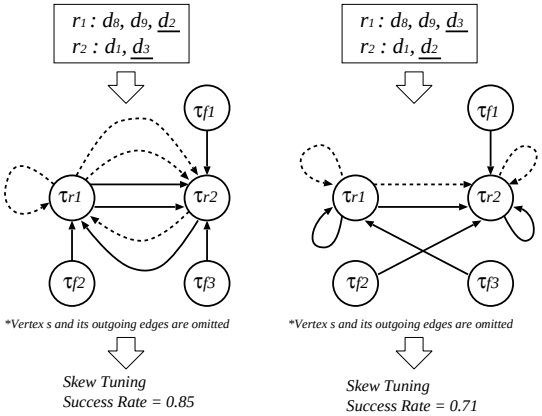


図 3.2: 製造後スキュー調整を考慮した資源割当結果

3.3　既存の資源割り当て手法

3.3.1　既存手法１：レフトエッジ

レフトエッジ法 [6] は、資源数最小を保証する資源割り当て手法である。まず、ライフタイムの開始ステップが１のデータや演算を１つ選択し、そのライフタイムの終了ステップ以降で最も早くライフタイムが開始するデータや演算を１つ選択し、同じ演算器、レジスタに割り当てする。こうした操作を実行ステップが終了するまで繰り返す。未割り当てのデータ、演算が残っている場合は再びステップ１から同様の操作を繰り返す。資源数制約は満たすがスキュー調整成功確率を考慮しない設計である。

図 3.3 は Defferential Equation のデータのライフタイムを表している。そして、図 3.4 がレフトエッジによるレジスタ割り当てである。まず開始時刻が早いデータから１つ取り出してレジスタの割り当てを決めていく。まず開始時刻が 1 であるデータ 1 をレジスタ 1 に割り当てる。すると、データ 1 のライフタイムは時刻 5 までなので時刻 6 以降の空きがある。そこで時刻 6 で始まるデータをレジスタ 1 で共有する。時刻 6 で始まるデータにデータ 12 があるので、データ 12 をレジスタ 1 に割り当てる。すると、データ 12 のライフタイムは時刻 7 までなので時刻 8 以降の空きがある。そこで時刻 8 で始まるデータ 15 をレジスタ 1 で共有する。ライフタイムが最後になったので次のレジスタの割り当てを実行する。開始時刻が 1 であるデータ 2 をレジスタ 2 に割り当てる。データ 2 のライフタイムが時刻 7 までなので時刻 8 以降の空きがある。しかし、時刻 8 以降で開始するデータがもう存在しないのでレジスタ 2 の割り当てを終了する。このように開始時刻が早いデータからレジスタに割り当て、ライフタイムの空きを無駄にしないようレジスタの共有を行う。

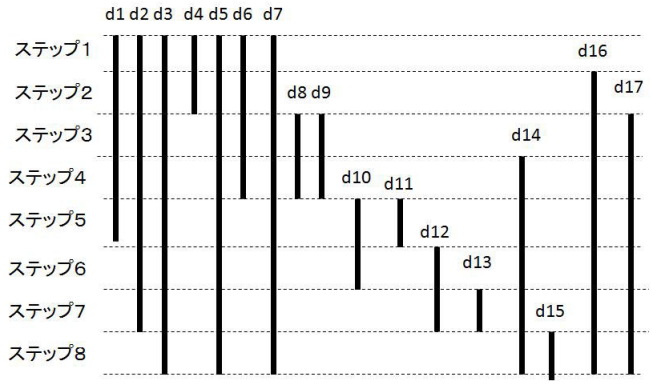


図 3.3: レフトエッジ：データのライフタイム

図 3.4: レフトエッジ：レジスタ割り当て

3.3.2 既存手法２：minTc

minTc は、スキュー値を製造前に調整して固定するタイプの回路を対象にした割り当て最適化手法 [3][4]（ここでは minTc と呼ぶ）が提案されている。製造前にスキュー値を調整するため遅延を定数としてスキューを考慮し、クロック周期を最小化する資源割り当てを出力する。

図 3.5 で minTc の説明をする。遅延を定数としてスキューを考慮するため、辺の重みは定数となり、ai+bi*Tc の形になる。まず、スキュー制約グラフにサイクルがあるとき１つのサイクルを構成する辺の総和が０になるクロック周期を計算する。サイクルが複数ある時、１つ１つのサイクルを構成する辺の総和が０になるクロック周期を計算する。すべてのサイクルの中でクロック周期が最大になる値が最小化されるように資源割り当てすることで、クロック周期最小化を達成する。最適化目標はスキュー調整成功確率の最大化とは異なるが、クロック周期最小化はこの目的のためにも有効に働くと考えられる。

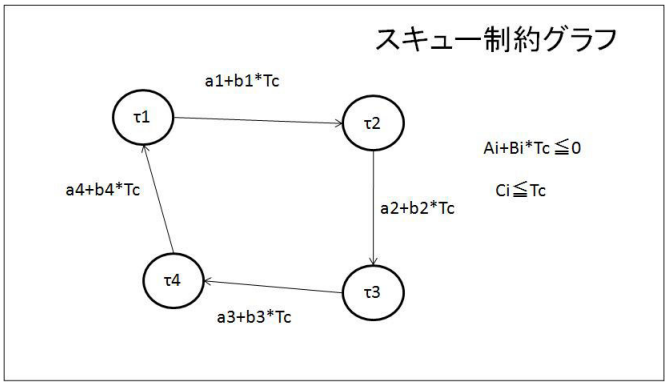


図 3.5: minTc

3.3.3 既存手法３：color

color は、スキュー値を製造後に調整するタイプの回路を対象とした資源割り当て手法[5]（ここでは color と呼ぶ）が提案されている。遅延が正規分布に従って変動すると想定して設計を行っている。平均遅延値におけるクロック周期を minTc で求めた最小クロック周期に制約し、スキュー制約グラフ上でサイクルを閉じる辺の正のなりやすさの総和を最小化する資源割り当てを出力する。この手法の問題点として、スキュー調整成功確率を直接的に評価できていないこと、整数線形計画法を用いるため計算量が大きいことなどがあげられる。

color は minTc のなかでスキュー調整成功確率が最大と考えられるものを選ぶ手法である。minTc ではクロック周期最小の資源割り当てが複数存在することがある。そのなかでスキュー調整成功確率が最大と考えられるものを選択する。

これらに対して、本研究ではより目的に近い定式化を目指し、スキュー調整成功確率を最大化する高位合成手法を検討、提案する。

第4章　スキュー調整成功確率の計算

3.1 節にて述べたようにスキュー調整成功確率は、スキュー制約グラフ上のすべてのサイクルを列挙し、すべてのサイクルの重みが負になる確率として求められる。

しかし一般の有向グラフにおいては 2 の頂点数乗個のサイクルが存在し得て、サイクル列挙に基づくスキュー調整成功確率計算は頂点数に対し、指数乗のオーダーの計算時間が必要になり、実用的とは言えない。スキュー調整成功確率の計算方法は重要な問題である。

4.1　辺の分類による評価

スキュー制約グラフに正重みサイクルが存在しない確率の計算を近似して評価する。以下の順序で評価する。

1. スキュー制約グラフ中の辺 1 辺ずつの正になる確率を計算する。
2. 辺が正になる確率が高いものから、非巡回有向グラフを形成する辺（FF 辺）に分類する。そのとき、非巡回有向グラフに足すとサイクルを形成する辺（FB 辺）が発生したら、サイクルを形成する辺として分類する。
3. サイクルを形成する辺 1 辺を含むサイクルを列挙し、正重みサイクルが存在しない確率を計算する。

辺分類後のスキュー制約グラフの例が図 4.1 である。辺の重みが正になる確率が高い順に e1,e2,e3...e7 とする。e1 から順に非巡回有向グラフを形成する辺に分類していく。すると、e6 を分類する時、サイクルを形成する辺が発生する。そのときは、e6 を FB 辺に分類し、操作を繰り返す。辺の分類が終わると FB 辺として e6,e7 が分類される。FB 辺 1 辺を含むサイクルを列挙し、サイクルの辺の重みの総和を計算し、正重みサイクルが存在しない確率を計算する。この手法による資源割り当て評価の問題点は、以下の近似を実行していることである。

- FB 辺を唯 1 辺含むサイクルのみ評価 ⇒ FB 辺 2 辺以上を含むサイクルもあり得る
- 異なる辺重みの分布が独立であると想定⇒ 異なる辺の重みの間に相関があり得る
- 異なるサイクルを独立に考えている ⇒異なるサイクルに含まれる辺の重みの間に相関、同一辺を共有するサイクルがあり得る

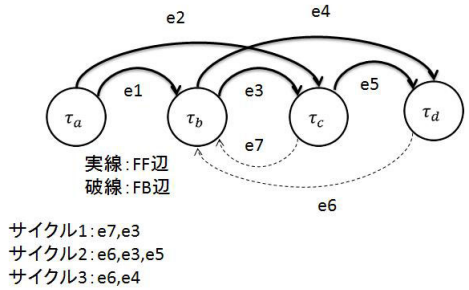


図 4.1: 辺分類後のスキュー制約グラフ

4.2 モンテカルロシミュレーションによる評価

スキュー制約グラフに正重みサイクルが存在しない確率の計算をモンテカルロシミュレーションで実行する。以下の順序で評価を行う。

1. 乱数を発生して、演算器の平均遅延と標準偏差をもとに遅延にセット
2. 辺重みが確定後、スキュー調整量計算
3. スキュー値が $0 \leqq \tau \leqq Tc$ かつ正サイクルが存在しないときスキュー調整成功

以上の計算を規定回数実行してスキュー調整成功確率の計算を行う。回数を増やせばシミュレーションの精度が上がるが、時間がかかる。今回は 10000 回実行して、確率の計算を行なっている。

図 4.2 は 2 手法の比較である。left edge による Defferential Equation の資源最小数資源割り当ての成功確率の比較を行った。辺の分類による評価はモンテカルロシミュレーションと誤差が生じている。この理由としては辺の分類による評価の問題点としてあげた近似が影響を与えていると考えられる。この誤差による資源割り当ての選択ミスはスキュー調整成功確率に多大な影響を与えるため、辺の分類による評価は使用せず、モンテカルロシミュレーションで評価を実行する。

4.3 モンテカルロシミュレーションの精度

モンテカルロシミュレーションは乱数を使う手法であり、その結果も決定論的なものではなく、統計論的不確実さが伴う。ここでは、モンテカルロシミュレーションにおける試行回数とシミュレーションの精度について検証する。以下の 3 種類の Dependence Graph で検証する。

- Defferential Equation

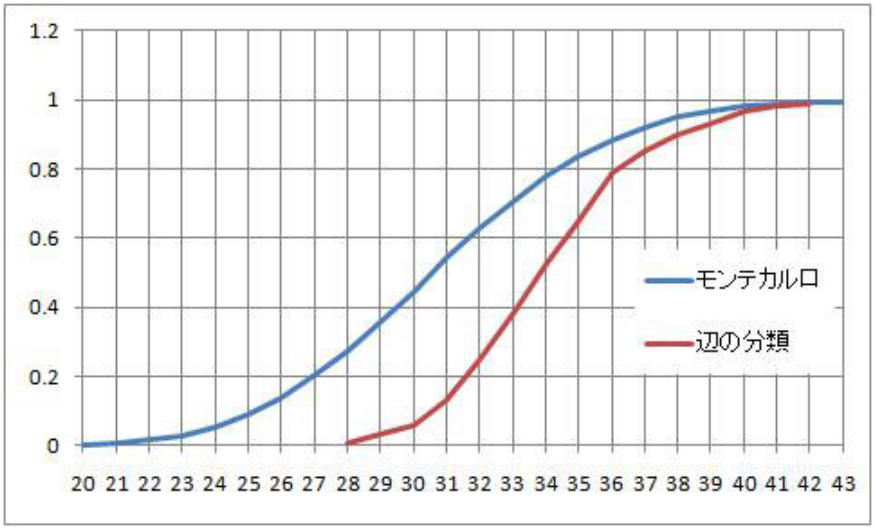


図 4.2: スキュー調整成功確率の計算

  - 演算数 10　データ数 17
  - 演算器最小数 3　レジスタ最小数 11

- Jaumann Wave Filter (ループを切り開いて、ＤＡＧ化したもの)
  - 演算数 17　データ数 29
  - 演算器最小数 3　レジスタ最小数 13

- Elliptic Wave Filter (但し、ループを切り開いて、DAG 化したもの)
  - 演算数 34　データ数 52
  - 演算器最小数 5　レジスタ最小数 19

入力するスケジュール済みデータフローグラフ、演算器情報、資源割り当ては付録に掲載する。資源割り当ては資源最小数の parallel rf を用いる。クロック周期は、変動の大きいと思われるスキュー調整成功確率が約 0.5 のクロック周期を入力する。Defferential Equation は 29ns、Jaumann Wave Filter が 35ns、Elliptic Wave Filter が 38ns を入力する。

同じ入力ファイルで 1000 回、3000 回、10000 回、30000 回、100000 回、300000 回、1000000 回実行を 20 回繰り返すことにより、スキュー調整成功確率のシミュレーションの回数と精度を検証する。その結果を以下に示す。図 4.3 が Defferential Equation、図 4.4 が Jaumann Wave Filter、図 4.5 が Elliptic Wave Filter の検証結果である。横軸が検証回数で、縦軸が調整成功確率である。繰り返し回数が多いほど精度が上がることが分かる。演算器数の多い Elliptic Wave Filter がほかの 2 つのグラフよりばらつきが大きいと想定

したが、10000 回の評価を行った 3 種類のグラフのなかでは 1 番ばらつきが小さかった。この理由の解明は今後の課題とする。

図の結果を参考に、必要となる検証の精度をその都度考えながら回数を決める必要がある。繰り返し回数による時間と値の精度を考慮し、今回は 10000 回で調整成功確率の評価を行い、資源割り当ての指標に使う。

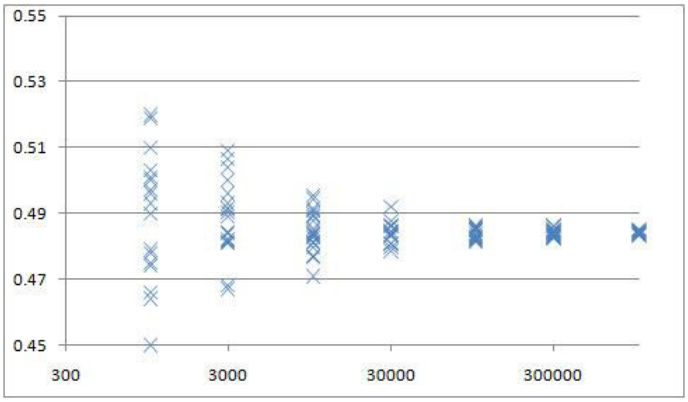


図 4.3: Defferential Equation

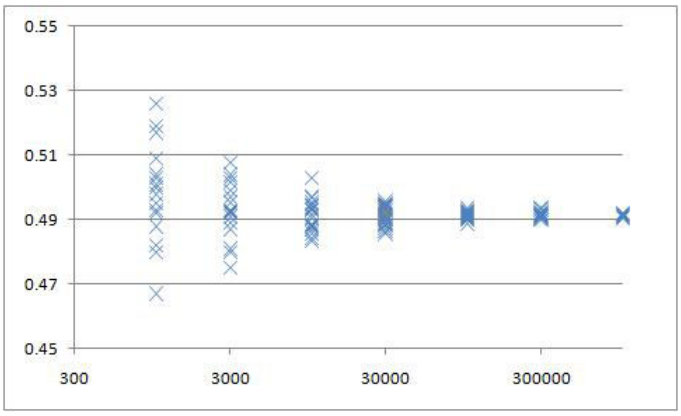


図 4.4: Jaumann Wave Filter

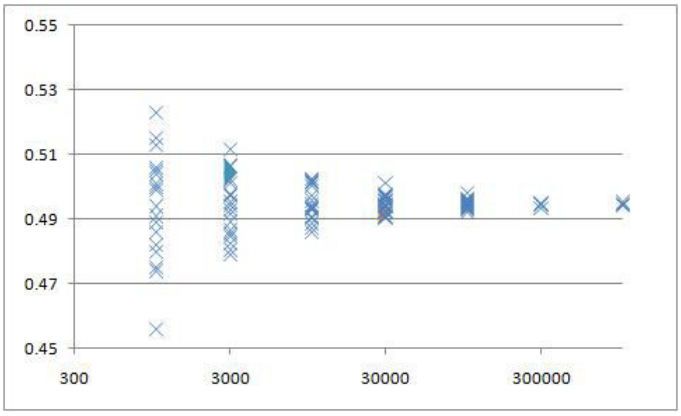


図 4.5: Elliptic Wave Filter

第5章　提案手法1：スキュー調整成功確率を考慮した資源割り当て手法

5.1　マージに基づく greedy な資源割り当て

3.4 で述べたレフトエッジはスキュー調整成功確率を考慮せず、minTc と color の 2 手法は直接スキュー調整成功確率を考慮していないので最適解を得られるとは限らない。今回本研究では、スキュー調整成功確率を直接評価して資源割り当てを決定する。可能性のあるすべての資源割り当て解を列挙して、それぞれに対してスキュー調整成功確率を評価し、その中の最善の解を求めれば最適解を得られる。しかし、すべての資源割り当てを列挙し評価することは解空間が大きくなりすぎ、現実的ではない。そこで、まずデータと演算をすべて異なるレジスタと演算器に割り当て初期解を用意する。その後、共有可能なデータと演算 2 つの組み合わせを列挙し、それぞれの組み合わせ後のスキュー調整成功確率を評価する。スキュー調整成功確率がもっとも大きい組み合わせを採用し、資源割り当てを更新する。2 つを 1 つに組み合わせる操作を繰り返し、資源を削減する。この手法を merge 法と呼ぶ。

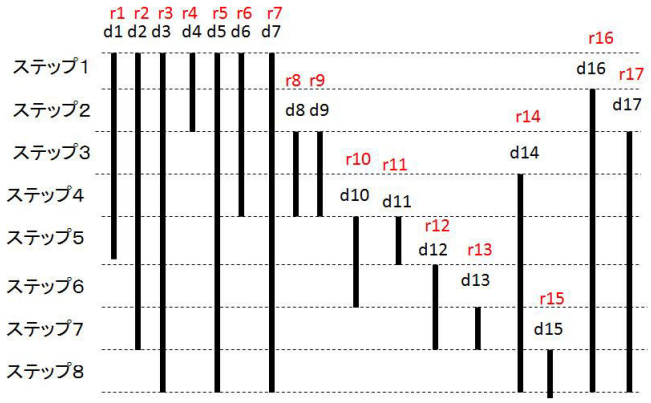


図 5.1: Defferential Equation のレジスタ割り当て初期解

具体的な流れをレジスタの共有によるレジスタの削減例を使い説明する。図 5.1 は Defferential Equation のデータのライフタイムである。まずデータをすべて異なるレジスタに割り当て初期解を得る。データのライフタイムが重ならない (共有可能な) 組み合わせ

共有レジスタ 1	共有レジスタ 2	共有後のスキュー調整成功確率
1	12	0.54
1	13	0.57
1	15	0.48
2	15	0.51
...	...	...
13	15	0.25

表 5.1: スキュー調整成功確率の値

を列挙する。レジスタ 1 と共有可能なレジスタはレジスタ 12,13,15 である。レジスタ 2 と共有可能なレジスタはレジスタ 15 である。レジスタ 3 と共有可能なレジスタはない。このようにすべての共有可能な組み合わせを列挙した後に、共有候補を 1 つ 1 つに対して、スキュー調整成功確率を評価する。すべての共有候補についてスキュー調整成功確率の評価が終わった後に、評価が最も良い共有を 1 つ選択し、資源割り当てを更新する。評価の値の例が表 5.1 である。

表の中で 1 番良い評価がレジスタ 1 とレジスタ 13 の共有ならば、割り当てを更新した後のレジスタのライフタイムは図 5.2 になる。この操作を共有可能なレジスタがなくなるまで行う。もしくは入力の資源数の上限を下回った時点でレジスタ割り当てを終了する。同様の操作を演算器割り当てについても行い、資源割り当て終了となる。割り当ての流れを図 5.3 に示す。

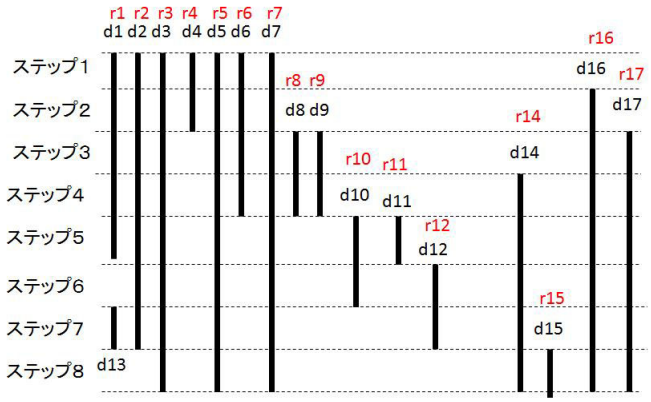


図 5.2: Defferential Equation のレジスタ割り当てレジスタ 1 つ削減後

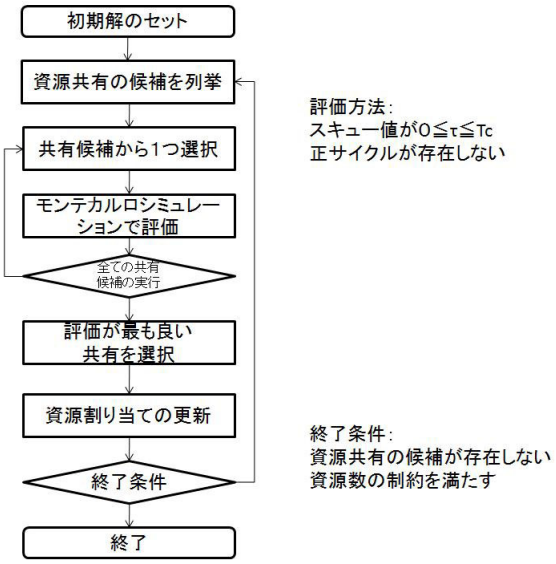


図 5.3: merge 法の設計フロー

5.2 実験と考察

入力するスケジュール済みデータフローグラフ、演算器情報は付録に掲載する。クロック周期は変動が大きいと思われるレフトエッジでスキュー調整成功確率が約 0.5 のクロック周期を入力する。Defferential Equation は 30ns、Jaumann Wave Filter は 35ns を入力する。資源数の上限は資源の最小数を入力する。

出力される資源割り当ては付録に掲載する。ここではレフトエッジ、minTc、color の３つの既存手法と merge 法を比較する。

merge 法による Defferential Equation での実験結果を図 5.4 に示す。横軸はクロック周期、縦軸は調整成功確率である。実験方法として、レジスタ割り当て実行後、演算器割り当てを行う方法 (merge rf)。演算器割り当て実行後、レジスタ割り当てを行う方法 (merge fr)。レジスタと演算器の割り当てを同時に行う方法 (merge mix)。の上記の 3 手法をとり、既存手法との比較を行った。3 手法とも資源数最小数での資源割り当てが可能であった。

次に、Jaumann Wave Filter での実験した。その結果を図 5.5 に示す。このグラフでのレジスタの最小数は 13 であるが、どれも最小数の解にはたどり着けなかった。rf,fr の方法ではレジスタ数 16、mix の方法ではレジスタ数 14 で限界となった。この理由として、ライフタイムを意識せずに資源削減を行ったことがあげられる。

merge mix の手法ではレジスタと演算器では演算器が先に共有されやすい傾向にあった。この理由としては、演算器を共有することで遅延のばらつきに相関が生まれることがあげられる。スキュー制約グラフ上でいうと辺の重みに相関関係が生まれ、サイクルの重みが正になりにくくなることが考えられる。そのため、スキュー調整成功確率が低くなるレジスタが後に共有されやすい傾向にあったと考えられる。

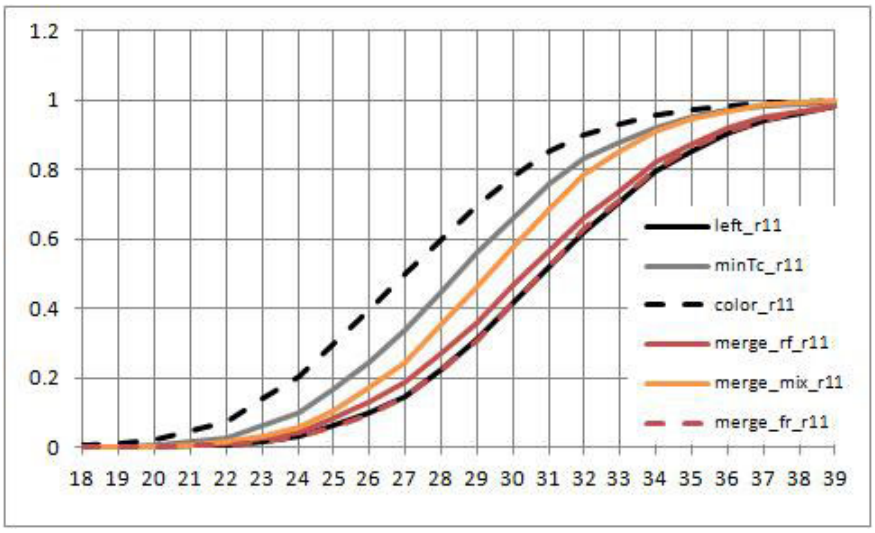


図 5.4: merge 法：Defferential Equation レジスタ数 11

merge 法の効果はレフトエッジより改善されているため認められるが十分とは言えない。Jaumann Wave Filter の資源数削減におけるスキュー調整成功確率の推移を図 5.6 に示す。横軸は資源数、縦軸は調整成功数である確率は徐々に下がらず、ある資源の共有によって格段に下がることが分かる。資源に余裕があるときは提案手法 1 の有効性が確認できる。しかし、資源数制約を満たせないという欠点が残る。そこで次の章で資源数制約を考慮した資源割り当てを行う。

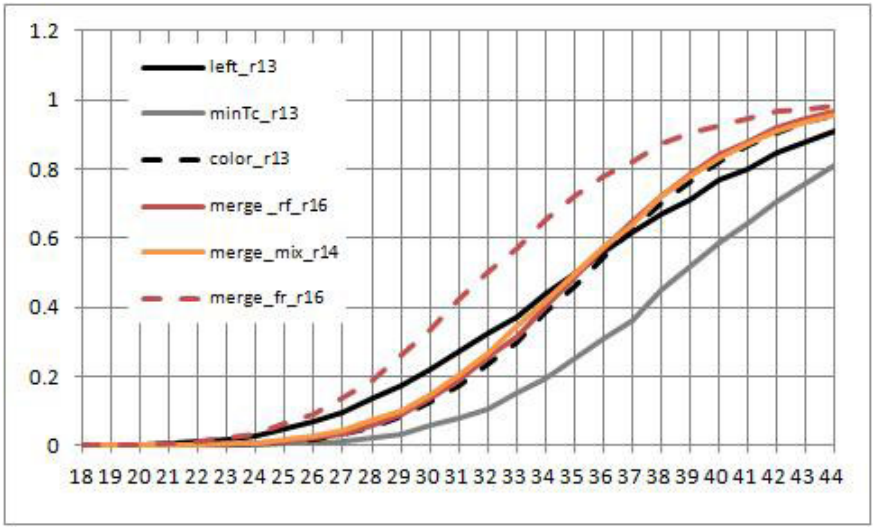


図 5.5: merge 法：Jaumann Wave Filter

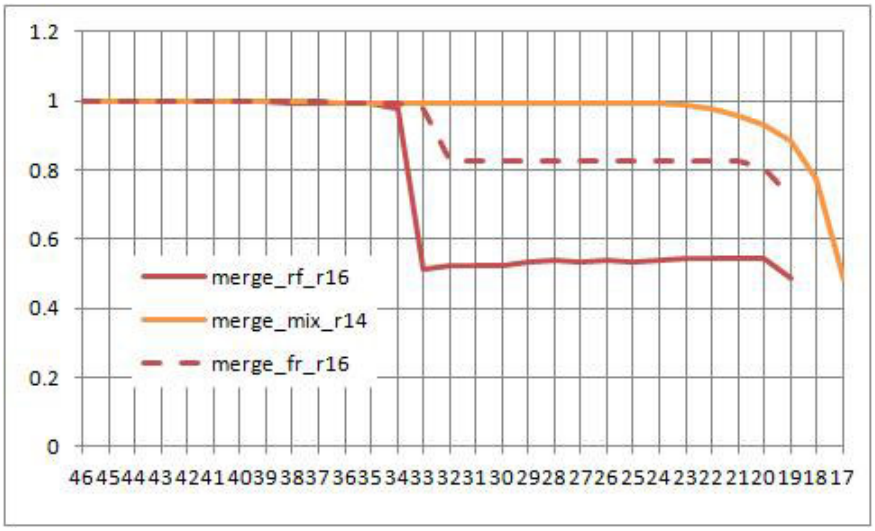


図 5.6: merge 法：Jaumann Wave Filter の調整成功確率の推移

第6章　提案手法2：資源数制約を満たす資源割り当て手法

6.1　並列レフトエッジ

merge 法では資源数制約を満たせない可能性があるので、並列レフトエッジでは資源数制約を満たしながらスキュー調整成功確率を考慮する資源割り当てを提案する。merge 法ではライフタイムを意識せずに資源削減を行ったので、資源数制約を満たせない可能性があった。並列レフトエッジではレフトエッジを応用した資源割り当てを行う。

通常のレフトエッジが常に 1 つの資源への割り当てを考えるのに対して、並列レフトエッジでは指定された資源のすべてに対して、同時並行的に割り当てを行うため、資源数制約を満たしながらスキュー調整成功確率を考慮する資源割り当てが可能である。

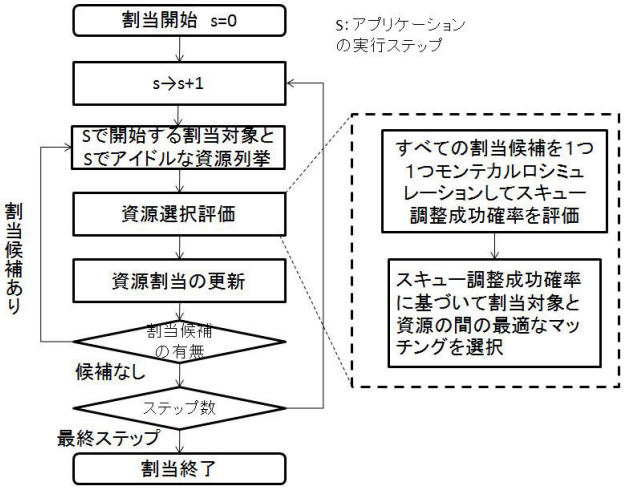


図 6.1: 並列レフトエッジの設計フロー

割り当ての流れを図 6.1 に示す。その手続きはスケジュールの第 1 実行ステップから順に、各実行ステップ s 毎に、s からライフタイムが始まる演算、データを s においてアイドルな（実行ステップ s で他の演算（データ）によって占有されていない）演算器、レジスタに割り当てを行うものである。1 つの実行ステップ s での割り当ての決定にあたっては、まず候補の演算（データ）の 1 つ 1 つに対して、アイドルな演算器 (レジスタ) の 1 つ 1 つに割り当てした際のスキュー調整成功確率を評価する。スキュー調整成功確率評価

の時には、未割り当ての演算、データは個別の演算器、レジスタに割り当てられてあものとして評価する。次いで、候補演算 (データ) 集合 Xs とアイドル演算器 (レジスタ) の集合 Ys を部集合とする完全 2 部グラフを構成し、演算ー演算器 (データーレジスタ) 辺に、対応する割り当てを行った際のスキュー調整成功確率を辺重みとして付加する。そうした上で、最小辺重み最大の Xs から Ys へのマッチング Ms を計算し、Ms に含まれる辺に対応する演算の演算器への (データのレジスタへの) 割り当てを採用する。こうした操作を実行ステップが終了するまで繰り返し、資源割り当てを終了する。この手法を並列レフトエッジと呼ぶ。

資源割り当ての際、最小辺重み最大の資源割り当てを選択する理由として、スキュー調整成功確率が一番低いサイクルが、全体のスキュー調整成功確率に最も影響を与えるためである。

6.2 例を使ったアルゴリズムの実行

実際にレジスタ最小数でレジスタ割り当てを行う例を示す。図 6.2 は Defferential Equation のデータのライフタイムである。実際の割り当ての流れを図 6.3、6.4、6.5、6.6 に掲載する。

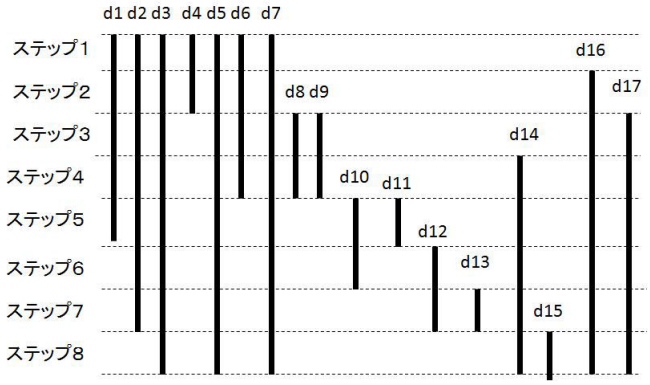
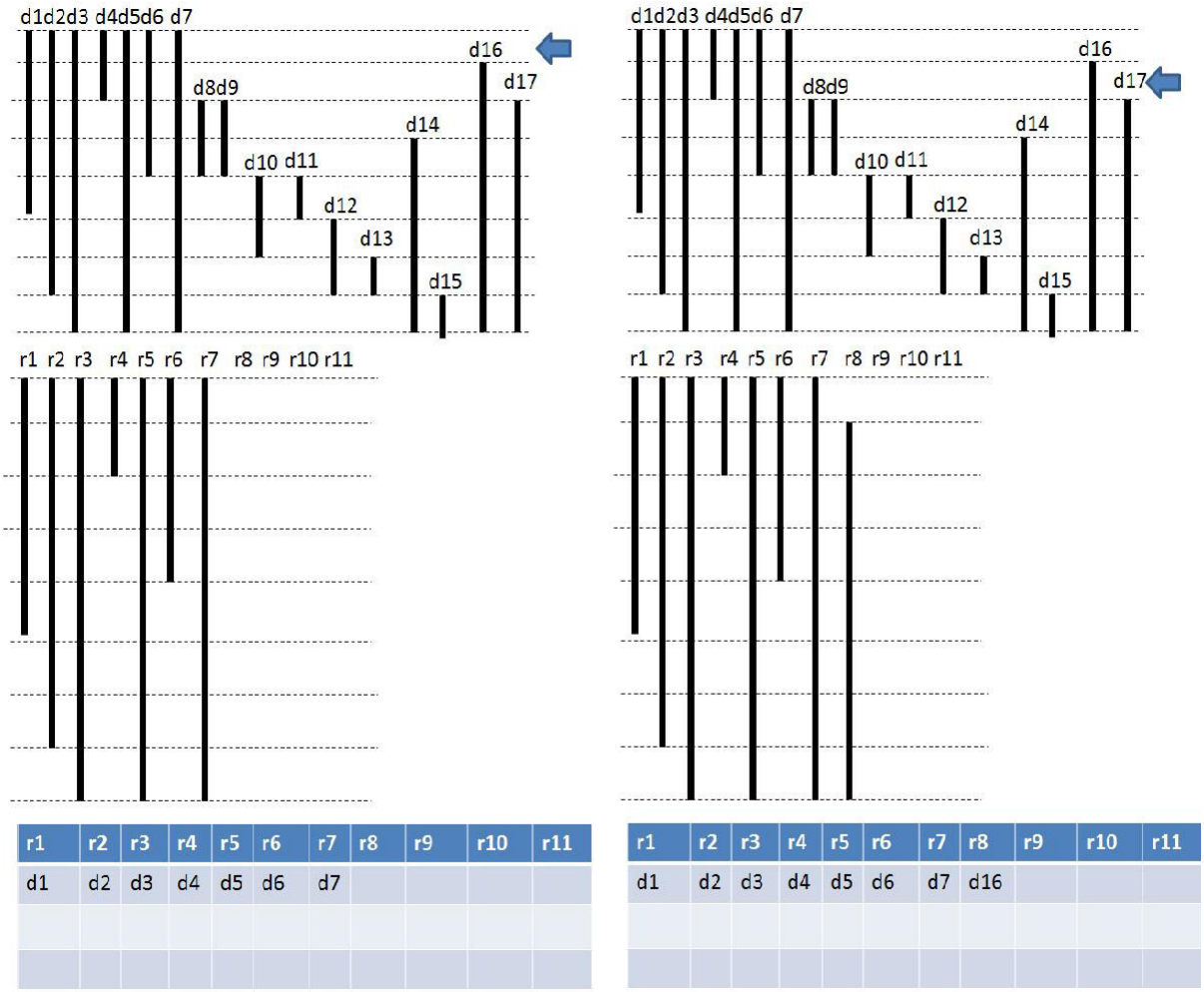


図 6.2: Defferential Equation のライフタイム

入力として、スケジュール済みデータフローグラフは付録の Defferential Equation。演算器情報は付録の演算器情報。クロック周期は 32ns。資源数は最小のレジスタ数 11。

初期解のスキュー調整成功確率は 0.994 である。開始ステップを 1 からはじめる。割り当て候補がデータ 1 から 7 であるが、アイドルなレジスタは存在しないので、データ 1 から 7 をレジスタ 1 から 7 に割り当て、開始ステップを 2 に更新する。

開始ステップ 2 のデータは 16 であるが、アイドルなレジスタがないのでデータ 16 をレジスタ 8 に割り当て、開始ステップを 3 に更新する。



r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11
d1	d2	d3	d4	d5	d6	d7				

r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11
d1	d2	d3	d4	d5	d6	d7	d16			

図 6.3: 並列レフトエッジのステップ 1 と 2

開始ステップ 3 のデータは 8，9，17 があるので割り当て候補は 3 つである。アイドルなレジスタは 4 で 1 つである。割り当て候補とアイドルなレジスタの組み合わせで最小値を最大化する組み合わせを求める。スキュー調整成功確率の値は

レジスタ 4 とデータ 8：0.993

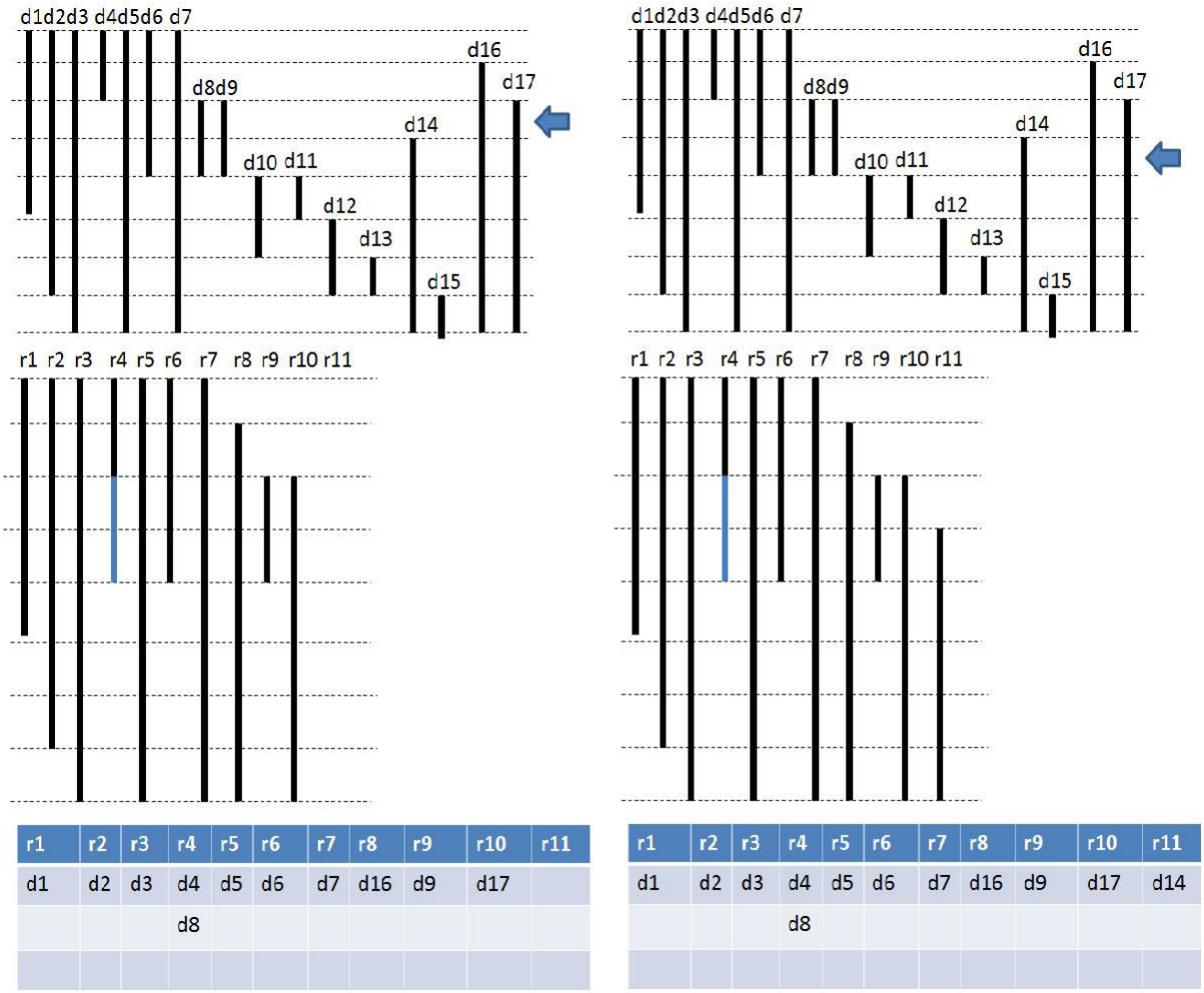
レジスタ 4 とデータ 9：0.916

レジスタ 4 とデータ 17：0.977

ということでレジスタ 4 とレジスタ 8 を共有させる。データ 9，17 はアイドルなレジスタがないので、データ 9，17 をレジスタ 9，10 に割り当て、開始ステップを 4 に更新する。

開始ステップ 4 のデータは 14 であるが、アイドルなレジスタは存在しないので、データ 14 をレジス 11 に割り当て、開始ステップを 5 に更新する。

開始ステップ 5 のデータは 10，11 であるので割り当て候補は 2 つである。アイドルなレジスタは 4，6，9 の 3 つである。割り当て候補とアイドルなレジスタの組み合わせで最



r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11
d1	d2	d3	d4	d5	d6	d7	d16	d9	d17	
			d8							

r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11
d1	d2	d3	d4	d5	d6	d7	d16	d9	d17	d14
			d8							

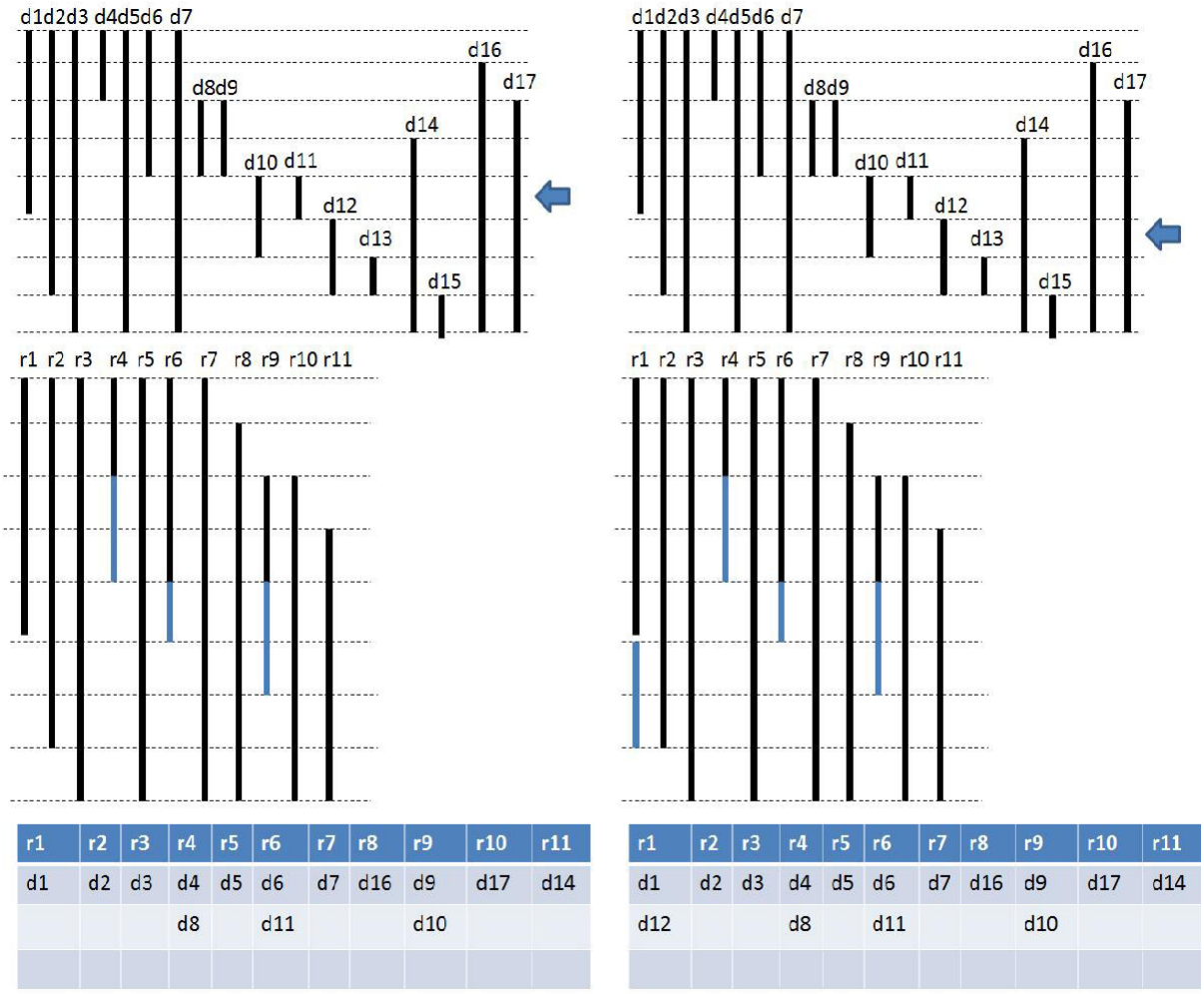
図 6.4: 並列レフトエッジのステップ 3 と 4

小値を最大化する組み合わせを求める。スキュー調整成功確率の値は

レジスタ 4 とデータ 10：0.991
レジスタ 4 とデータ 11：0.991
レジスタ 6 とデータ 10：0.992
レジスタ 6 とデータ 11：0.993
レジスタ 9 とデータ 10：0.992
レジスタ 9 とデータ 11：0.911

ということでレジスタ 9 とデータ 10 を共有させる。すると割り当て候補がデータ 11、アイドルなレジスタが 4,6 となる。割り当て候補とアイドルなレジスタの組み合わせで最小値を最大化する組み合わせを求める。スキュー調整成功確率の値は

レジスタ 4 とデータ 11：0.990
レジスタ 6 とデータ 11：0.991



r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11
d1	d2	d3	d4	d5	d6	d7	d16	d9	d17	d14
			d8		d11			d10		

r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11
d1	d2	d3	d4	d5	d6	d7	d16	d9	d17	d14
d12			d8		d11			d10		

図 6.5: 並列レフトエッジのステップ５と６

ということでレジスタ 6 とデータ 11 を共有させる。割り当て候補がなくなったので開始ステップを 6 に更新する。

開始ステップ 6 のデータは 12 であるので割り当て候補は１つである。アイドルなレジスタは 1,4,6 の３つである。割り当て候補とアイドルなレジスタの組み合わせで最小値を最大化する組み合わせを求める。スキュー調整成功確率の値は

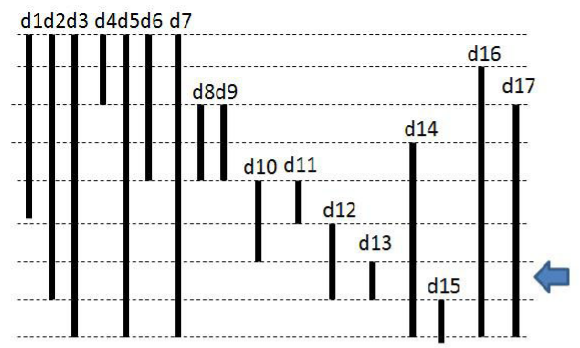
レジスタ 1 とデータ 12：0.881

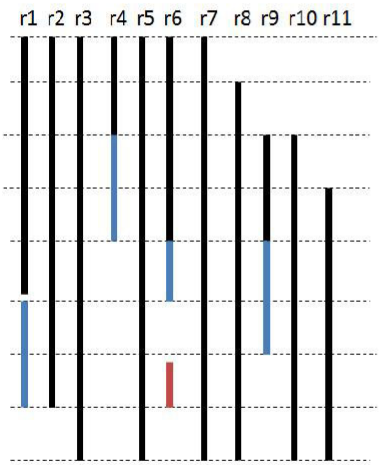
レジスタ 4 とデータ 12：0.671

レジスタ 6 とデータ 12：0.157

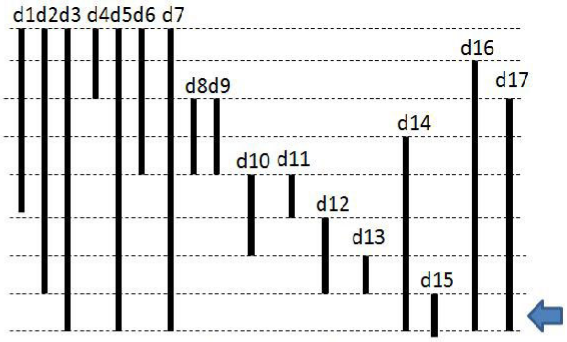
ということでレジスタ 1 とデータ 12 を共有させる。割り当て候補がなくなったので開始ステップを 7 に更新する。

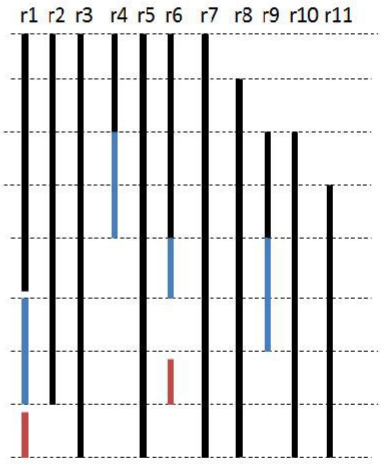
開始ステップ 7 のデータは 13 なので割り当て候補は１つである。アイドルなレジスタは 4,6,9 の３つである。割り当て候補とアイドルなレジスタの組み合わせで最小値を最大





r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11
d1	d2	d3	d4	d5	d6	d7	d16	d9	d17	d14
d12			d8		d11			d10		
					d13					





r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11
d1	d2	d3	d4	d5	d6	d7	d16	d9	d17	d14
d12			d8		d11			d10		
d15					d13					

図 6.6: 並列レフトエッジのステップ 7 と 8

化する組み合わせを求める。スキュー調整成功確率の値は

レジスタ 4 とデータ 13：0.864

レジスタ 6 とデータ 13：0.873

レジスタ 9 とデータ 13：0.807

ということでレジスタ 6 とデータ 13 を共有させる。割り当て候補がなくなったので開始ステップを 8 に更新する。

開始ステップ 8 のデータは 15 なので割り当て候補は 1 つである。アイドルなレジスタは 1,2,4,6,9 の 5 つである。割り当て候補とアイドルなレジスタの組み合わせで最小値を最大化する組み合わせを求める。スキュー調整成功確率の値は

レジスタ 1 とデータ 15：0.808

レジスタ 2 とデータ 15：0.793

レジスタ 4 とデータ 15：0.581

レジスタ 6 とデータ 15：0.141
レジスタ 9 とデータ 13：0.755
ということでレジスタ 1 とデータ 15 を共有させる。割り当て候補がなくなり、ステップ 8 が最後のステップなのでレジスタの割り当て終了となる。
この流れを演算器でも行う。実験方法として、レジスタ割り当て実行後、演算器割り当てを行う方法 (parallel rf)。演算器割り当て実行後、レジスタ割り当てを行う方法 (parallel fr)。レジスタと演算器の割り当てを同時に行う方法 (parallel mix) の 3 手法をとり、既存手法との比較を行う。

6.3 並列レフトエッジの資源量保証

ここで並列レフトエッジの定理を示し、証明する。

定理 6.3.1 サイクルのないアルゴリズムにおいて並列レフトエッジはライフタイムの最大重なり以上の指定の資源数で割り当てが可能である。

資源数制約の最小値を M_o、資源数制約を $M(M \geq M_o)$ とする。並列レフトエッジでは実行ステップ 1 から順にそのステップでライフタイムが始まる演算 (データ) の演算器 (レジスタ) への割り当てを行う。実行ステップ i から始まる演算 (データ) の個数を K_i とする。明らかに $K_i \leq M_o$
実行ステップ 1 では全く未割り当ての演算器（レジスタ）を割り当てすることになり、$M \geq M_o \geq K_i$ より常に可能。
次に実行ステップ 1 から $s-1$ まで同じように割り当てが出来たものとして、実行ステップ s での割り当てを考える。ここで、実行ステップ s が空状態になっている演算器は $s+1$ 以降も空状態になっている。s における空状態の演算器の個数を L_s とすると $M - L_s$ 個の演算器はステップ s を演算で使っている。この時、ステップ s でのライフタイムの重なりは、$K_s + (M - L_s)$ であり、$K_s + (M - L_s) \leq M_o \leq M$ よって、$L_s \geq K_s$
従って、s から始まる K_s 個の演算は s において空である L_s 個の演算器に割り当てできる。
従って、最初の実行ステップから最後の実行ステップまで、すべてのステップで割り当てができて、すべての演算 (データ) を M 個の資源に割り当てできる。

6.4 実験と考察

入力するスケジュール済みデータフローグラフ、演算器情報は付録に掲載する。クロック周期は変動が大きいと思われるレフトエッジでスキュー調整成功確率が約 0.5 のクロック周期を入力する。Defferential Equation は 30ns、Jaumann Wave Filter は 35ns、Elliptic Wave Filter は 38ns を入力する。資源数の上限は資源の最小数を入力する。

出力される資源割り当ては付録に掲載する。Defferential Equation と Jaumann Wave Filter でレフトエッジ、minTc、color の３つの既存手法と merge 法と並列レフトエッジを比較する。また、Elliptic Wave Filter でレフトエッジ、minTc、color の３つの既存手法と並列レフトエッジを比較する。

並列レフトエッジによる Defferential Equation の資源数最小数での実験結果を図 6.7 に示す。横軸はクロック周期、縦軸は調整成功確率である。

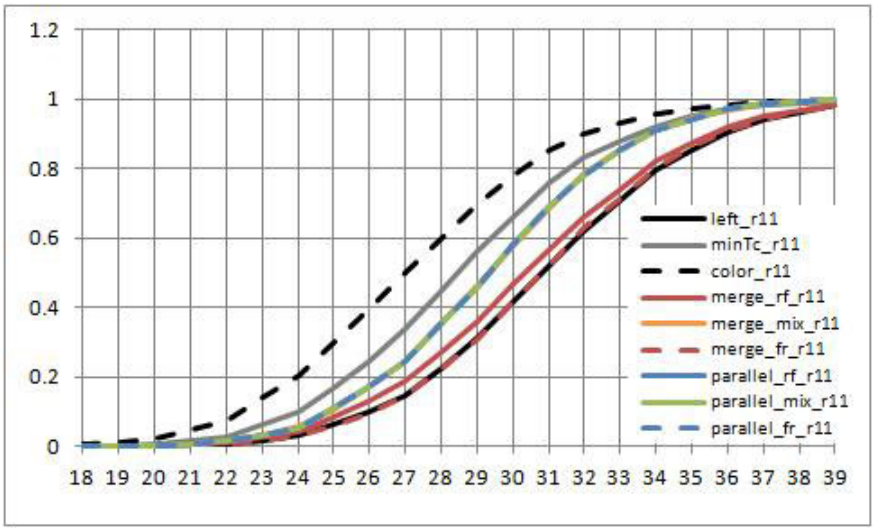


図 6.7: 並列レフトエッジ：Defferential Equation レジスタ数 11

次に Jaumann Wave Filter の資源数最小数での実験結果を図 6.8 に示す。さらに、レジスタ数を１つ増やした実験結果が図 6.9 である。

Elliptic Wave Filter の資源数最小数での実験結果を図 6.10 に示す。さらに、レジスタ数を１つ増やした実験結果が図 6.11 である。

また、minTc、color、並列レフトエッジの計算時間の比較を行った。計算環境として、minTc と color は altix-xe-01（Quadcore Intel Xeon 2.8GHz (8core)）で実行し、並列レフトエッジは AMD Opteron(2GHx(2core)) で実行した。その結果を表 6.1 に示す。縦軸は実験した回路とレジスタの制約数、横軸は手法である。() は打ち切りによる暫定解で結果を検証したことを表す。

並列レフトエッジの計算時間の多くはモンテカルロシミュレーションで使われるので、Defferential Equation のように回路の規模 (解空間) が小さいときは minTc のほうが計算時間が速い。しかし、Elliptic Wave Filter のように回路の規模 (解空間) が大きくなると並列レフトエッジのほうが速いものが発生する。

parallel mix の手法ではレジスタと演算器ではレジスタが先に共有されやすい傾向にあった。この理由としては、演算器を共有することで遅延のばらつきに相関が生まれることがあげられる。スキュー制約グラフ上でいうと辺の重みに相関関係が生まれ、サイクルの重

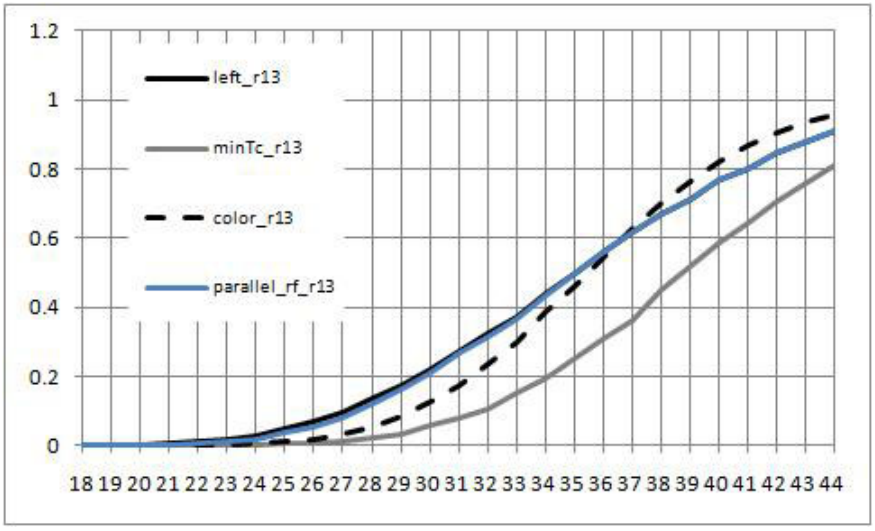


図 6.8: 並列レフトエッジ：Jaumann Wave Filter　レジスタ数 13

	minTc	color	parallel rf
Defferential Equation r11	0.33[s]	2858.23[s]	15[s]
Jaumann Wave Filter r13	23.42[s]	(7200[s])	40[s]
Jaumann Wave Filter r14	101.73[s]	–	50[s]
Elliptic Wave Filter r19	1200[s]	(7200[s])	1200[s]
Elliptic Wave Filter r20	3600[s]	–	1500[s]

表 6.1: 各種法による計算時間

みが正になりにくくなることが考えられる。そのため、スキュー調整成功確率が低くなるレジスタが先に共有されやすい傾向にあったと考えられる。

並列レフトエッジも最適解を得られる保証はない。しかし、merge 法と違い資源数制約を満たす資源割り当てが可能である。また、資源数が同じ時は並列レフトエッジは merge 法より良い解が出力された。

レフトエッジ、minTc、color の 3 つの並列レフトエッジを minTc と color の 2 つの既存手法を比較し、Defferential Equation と Jaumann Wave Filter で minTc と color のほうが良い解を出力した。この理由として、資源割り当ての際すべての演算とデータの情報を同時に見ていないため、最適解が出力されない可能性がある。全体を見て資源割り当てする手法が今後必要になる。

そんな中、Elliptic Wave Filter の資源数最小数での資源割り当てで既存手法を上回る資源割り当てが出力された。この理由として、回路規模が大きくなることで実際のスキュー

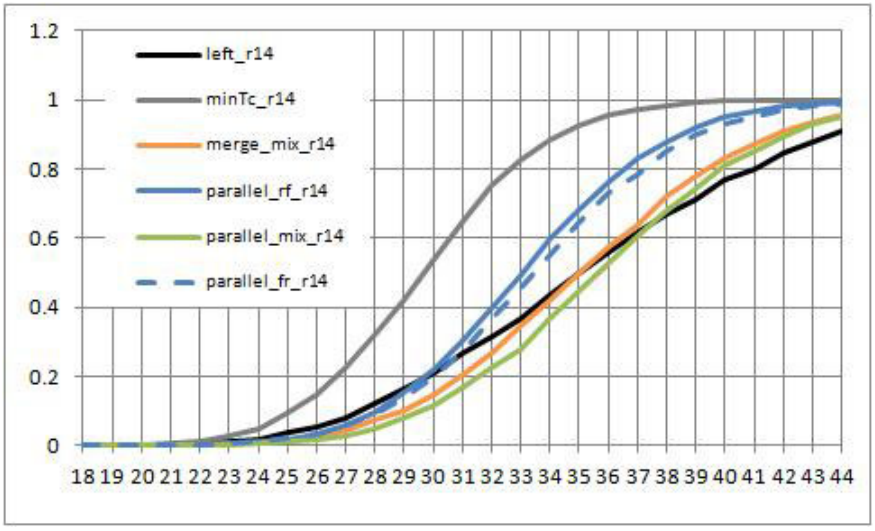


図 6.9: 並列レフトエッジ：Jaumann Wave Filter　レジスタ数 14

調整成功確率をもとに資源割り当てをしている効果が表れたと考えられる。

6.5　資源割り当ての繰り返し改善

並列レフトエッジの解をもとによりよい解を求める方法を考える。その流れを以下に示す。

まず、並列レフトエッジの解を初期解として持つ。次に、指定のライフタイム内でおさまるデータと演算を割り当て候補とし、割り当て候補と共有可能な共有候補を列挙する。その後、提案手法 2 同様にすべての割り当て候補と共有候補の組み合わせで評価を行い、最小値を最大化する資源割り当てを行う。指定のライフタイム内の資源割り当てがすべて終了したら、元の資源割り当てと比較する。元の資源割り当てよりスキュー調整成功確率が向上するなら資源割り当てを更新する。

この資源の変更をステップ 1 から 1、ステップ 1 から 2、ステップ 2 から 2…とすべてのライフタイムで実行する。資源割り当ての更新が行われたらまたステップ 1 から 1 繰り返し実行する。すべてのライフタイムで評価を行い、元の資源割り当てよりスキュー調整成功確率が向上しなくなったら終了する。

資源割り当ての繰り返し改善の例を Defferential Equation を用いて図で説明する。入力するスケジュール済みデータフローグラフ、演算器情報、クロック周期は 6.4 節同様である。入力する資源割り当ては付録の parallel rf r11 を使用し、レジスタ割り当ては図である。図は指定のライフタイムがステップ 6 から 7 の時のレジスタ割り当て例である。指定のライフタイム内におさまるデータはデータ 12 とデータ 13 である。割り当て候補になるデータ 12 と 13 を共有できるレジスタはレジスタ 1，4，6 である。その時の値を並列レフ

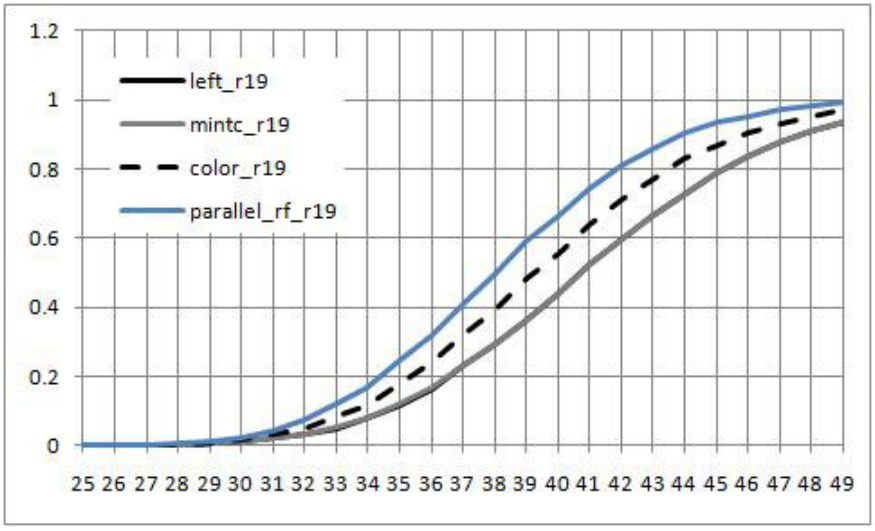


図 6.10: 並列レフトエッジ：Elliptic Wave Filter　レジスタ数 19

トエッジ同様最小値を最大化する割り当てを選択し、割り当てを更新する。割り当て候補をすべて共有させたときの値がもとの資源割り当てよりスキュー調整成功確率が向上するなら割り当てを更新し、新しいスキュー調整成功確率がもとの資源割り当てを下回るならもとの資源割り当てに戻す。

6.6　資源割り当ての繰り返し改善の実験と考察

今回はレジスタの変更を行った後、演算器の変更。そのあとに再びレジスタの変更という順番で双方がスキュー調整成功確率が向上しなくなったら終了した。

入力するスケジュール済みデータフローグラフ、演算器情報は付録に掲載する。Defferential Equation は 30ns、Jaumann Wave Filter は 35ns、Elliptic Wave Filter は 38ns を入力する。資源数は Defferential Equation は資源数最小、Jaumann Wave Filter は資源数最小からレジスタを１つ増やしたレジスタ 14、Elliptic Wave Filter も資源数最小からレジスタを１つ増やしたレジスタ 20 を入力する。入力する資源割り当ては付録の parallel rf を用いる。

出力される資源割り当ては付録に change という手法名で掲載する。Defferential Equation、Jaumann Wave Filter、Elliptic Wave Filter でレフトエッジ、minTc、color の３つの既存手法と並列レフトエッジと資源割り当ての繰り返し改善を比較する。

図 6.14 が Defferential Equation の資源数最小での結果。図 6.15Jaumann Wave Filter の資源数最小からレジスタ数を１つ増やした結果。図 6.16Elliptic Wave Filter の資源数最小からレジスタ数を１つ増やした結果である。

Jaumann Wave Filter と Elliptic Wave Filter の資源数最小はライフタイムの空きが少

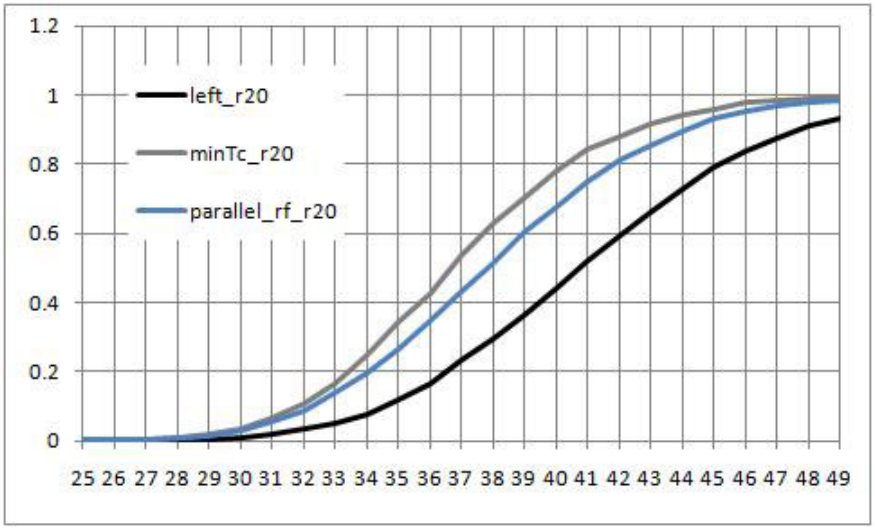


図 6.11: 並列レフトエッジ：Elliptic Wave Filter　レジスタ数 20

ないため改善する解が得られなかった。資源割り当ての繰り返し改善によるスキュー調整成功確率の向上がみられた。しかし、既存手法に近づきはしたが既存手法よりスキュー調整成功確率が高くなることはなかった。

資源割り当ての繰り返し改善の問題点として、ローカルな最適解に陥っていることが考えられる。改善方法としてルールを設けてローカルな最適解をさける割り当てが必要である。

ルールを設ける基準としてライフタイムとスキュー調整成功確率の関係を利用する方法が考えられる。資源共有することで調整成功確率が下がる理由として、セットアップ、ホールド条件が厳しくなることとスキュー値が同じになることが考えられる。前者はライフタイムをあけることで効果があるが、後者はライフタイムを空けても効果はない。この情報をうまく利用してルールを設け、ローカルな最適解を避ける割り当てを検討したい。

図 6.12: 資源割り当ての繰り返し改善で入力するレジスタ割り当て

図 6.13: ライフタイムがステップ 6 から 7 の資源割り当ての繰り返し改善による評価

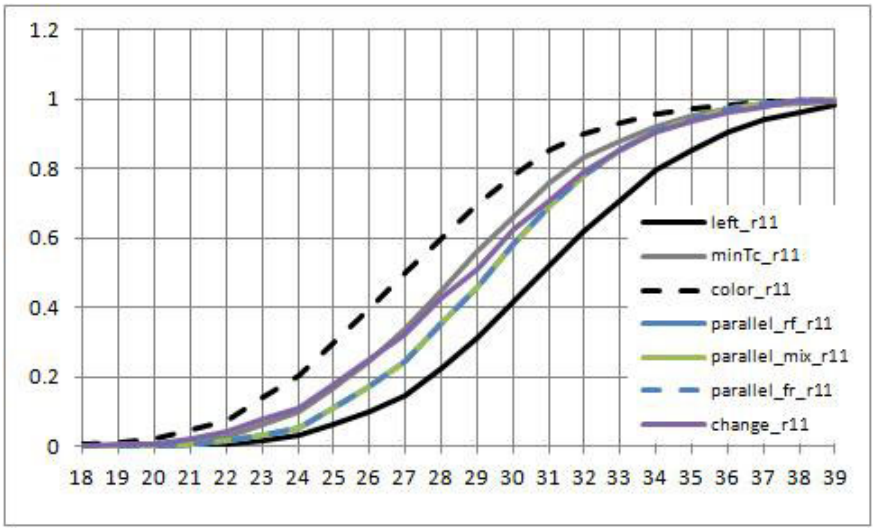


図 6.14: 資源割り当ての繰り返し改善：Defferential Equation　レジスタ数 11

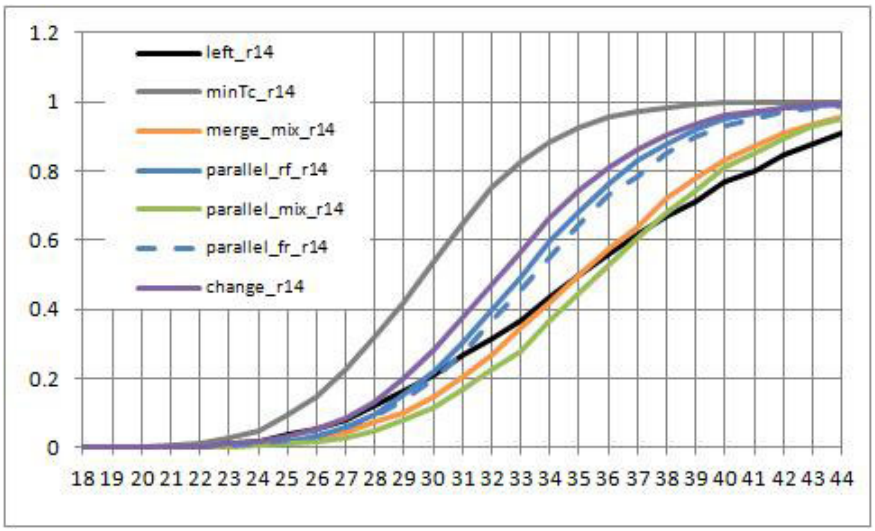


図 6.15: 資源割り当ての繰り返し改善： Jaumann Wave Filter　レジスタ数 14

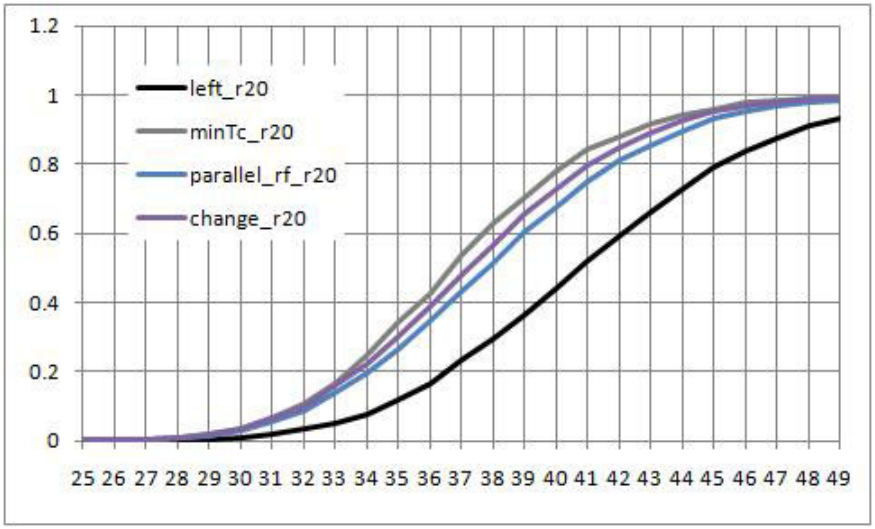


図 6.16: 資源割り当ての繰り返し改善：Elliptic Wave Filter　レジスタ数 20

第7章　まとめ

本研究では製造ばらつきに対して、この製造後スキュー調整にて高性能な集積回路を高い歩留まりで製造するための設計手法を検討、提案した。回路を正常動作させるために、データの取り込みを確実に行うタイミング制約であるセットアップ条件とホールド条件をすべての演算について書きだし、それらすべてを同時に満足するような各レジスタと演算器のタイミングスキュー値（クロック信号からの個別の遅れ時間）を決定する必要がある。タイミングスキュー値の決定は、１つ１つのタイミング制約を有向辺とするスキュー制約グラフ上で特別に定めた始点からすべての頂点への最長パス長を求めることによってなされる。このことから、遅延ばらつきのもとで回路を正常動作させる確率（スキュー調整成功確率）の最大化は、スキュー制約グラフ上にある全サイクルの辺重みの総和が負になる確率の最大化に等しい。また、高位合成は動作記述からレジスタ転送レベル回路記述に変換する工程であり、回路中の信号の流れとそのタイミングを決める重要な設計段階である。

本研究で 2 つの資源割り当て手法を提案した。merge 法はスキュー調整成功確率を考慮した資源割り当て手法であり、並列レフトエッジは資源数制約を満たしながらスキュー調整成功確率を考慮した資源割り当て手法である。

merge 法はまず、すべてのデータと演算を共有させずに資源割り当てし、初期解を得る。そこから共有可能な資源のすべての組み合わせでスキュー調整成功確率を計算し、スキュー調整成功確率を最大化する資源割り当てを選択する。計算と選択を繰り返し、資源数制約を満たすか共有可能な資源がなくなるまで続ける。

並列レフトエッジは merge 法で不可能だった資源数制約を満たすため、データと演算のライフタイムを使い、資源割り当てを進める。並列レフトエッジも merge 法同様の初期解を得る。そこから、データと演算の開始ステップの早いものから資源割り当てを行う。指定のライフタイムを開始ステップとするデータと演算を資源共有可能な資源との組み合わせでスキュー調整成功確率を計算し、スキュー調整成功確率の最小値を最大化する資源割り当てを選択する。計算と選択を繰り返し、開始ステップが最後のステップになるまで続ける。

既存手法と提案手法について、回路合成実験を行ったところ、並列レフトエッジ法は通常のレフトエッジ法に対して常に優れた解を得るものの、minTc、color 手法に対しては、幾つかの例外を除いて、その解は劣るものであった。これは、並列レフトエッジ法の大域的最適解計算能力の低さが主因と考えられる。その一方で、並列レフトエッジ法が minTc、color よりもすぐれた解を得ている例もあり、スキュー調整成功確率を正しく評

価しながら設計を行うことの重要性が示された。

今後、スキュー調整成功確率評価の正しさを継承する、より優れた解探索手法の開発が望まれる。また、モンテカルロシミュレーションの精度で考察した演算器数とばらつきの関係の解明やタイミングスキュー調整を考慮したスケジューリング、演算器の遅延の分布を把握するための統計的遅延解析の採用も今後の課題である。

参考文献

[1] P. S. Zuchowski, P. A. Habitz, J. D. Hayes and J. H. Oppold, “Process and environmental variation impacts on asic timing”, Proc. International Conference on Computer Design, pp. 336-342(2004).

[2] J. Singh and S. Sapatnekar, “Statistical timing analysis with correlated non-gaussian parameters using independent component analysi”, In Proc. ACM/IEEE DAC, pp. 155-160(2006).

[3] S.-H.Huang, C.-H.Cheng, Y.-T.Nieh, W.-C.Yu, “Register Binding for Clock Period Minimization”, Proc.DAC 2006, pp439-444.

[4] M.kaneko, “A Complete Framework of Simulaneous Functional Unit and Register Binding with Skew Scheduling”, Proceedings of IEEE International Symposium on Quality Electronic Design, pp.189-195, 2011.

[5] M.Kaneko and K.Inoue, “Ordered Coloring-Based Resourc Binding for Datapaths with Improved Skew-Adjustability”, Proceedings of ACM Great Lakes Symposium on VLSI, pp.307-312, 2011.

[6] Petra Michel, Ulrich Lauther, and Peter Duzy, The Synthesis Approach To Digital System Design, Kluwer Academic Pub, 1992, pp.197-199.

付 録A　実験の詳細

A.1　スケジュール済みデータフローグラフ

３つのスケジュール済みデータフローグラフを示す。図 A.1 が Defferential Equation のスケジュール済みデータフローグラフ、図 A.2 が Jaumann Wave Filter のスケジュール済みデータフローグラフ、図 A.3 が Elliptic Wave Filter のスケジュール済みデータフローグラフである。

図の番号はデータ番号を表し、O に続く番号が演算番号を表す。

図 A.2 と図 A.3 については、演算器の出力のデータ番号と演算番号が一致している。

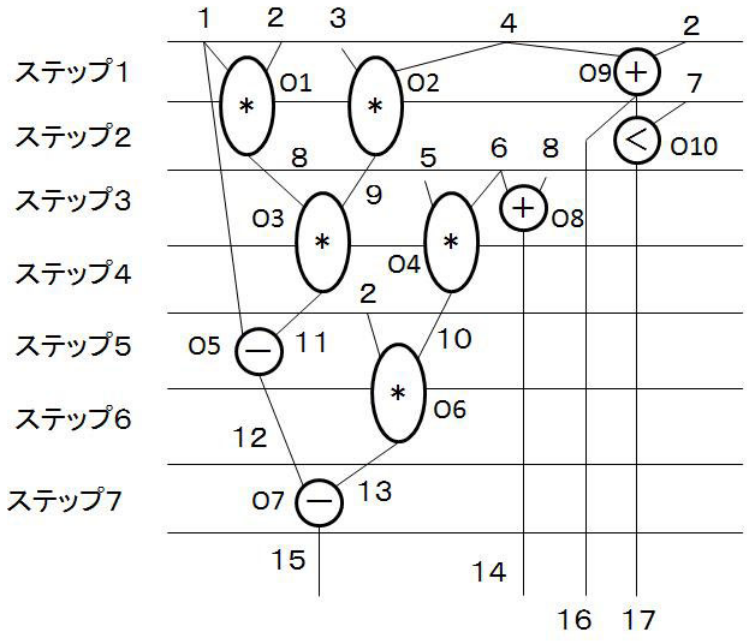


図 A.1: Defferential Equation

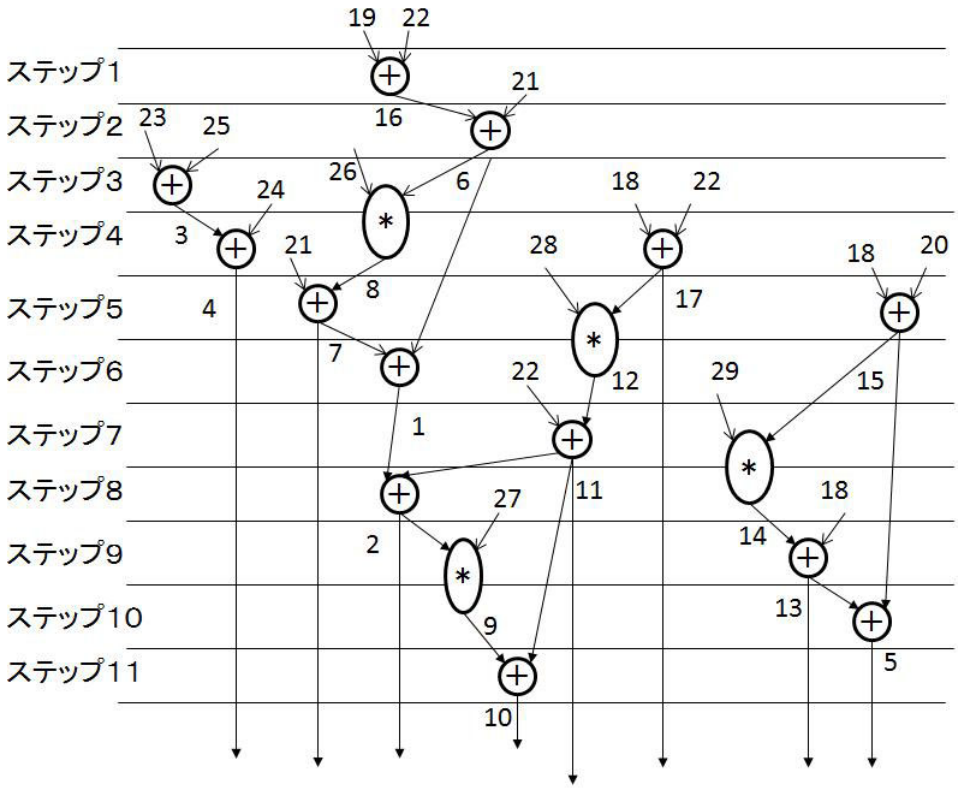


図 A.2: Jaumann Wave Filter

A.2　実験で使用した演算器情報

表 A.1 は実験で使用した演算器情報である。演算器のタイプ、機能、遅延の平均値と標準偏差を示している。

A.3　各実験における資源割り当て結果

表 A.2 から A.7 はデータを格納するレジスタと演算を行う演算器の割り当て情報を表示している。横軸はどのレジスタ、演算器かという情報。縦軸はどの手法で導いたかという情報とレジスタ数を表している（手法名　r ○○）。表内部の数字がどのデータ、演算かという情報である。

演算器タイプ	機能	最大遅延平均値	最大遅延標準偏差	最小遅延平均値	最小遅延標準偏差
1	*	50	10	15	3
2	+	35	7	12	2.4
	-	40	8	12	2.4
	C	14	2.8	12	2.4

表 A.1: 演算器情報

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11
left r11	1,12,15	2	3	4,8,10,13	5	6	7	14	9,11	16	17
minTc r11	1	2	3	8,10,13	5	6,12,15	7	17	4,9,11	14	16
color r11	1	2	3	4,8,10	5	6,12,15	7	17	9,11,13	14	16
merge rf r11	1,12,15	2	3	4,8,13	5	6,10	7	14	9,11	16	17
merge mix r11	1,12,15	6,11,13	9,10	4,8	2	3	5	7	14	17	16
merge fr r11	1,12,15	4,8,11,13	6,10	2	3	5	7	9	14	17	16
parallel rf r11	1,12,15	2	3	4,8	5	6,11,13	7	17	9,10	14	16
parallel mix r11	1,12,15	2	3	4,8	5	6,11,13	7	17	9,10	14	16
parallel fr r11	1,12,15	2	3	4,8	5	6,11,13	7	17	9,10	14	16
change r11	1,12,15	2	3	8	5	6,11,13	7	4,17	9,10	14	16

表 A.2: Defferential Equation のレジスタ割り当て

	F1(type1)	F2(type1)	F3(type2)
left r11	1,3,6	2,4	5,7,8,9,10
minTc r11	1,3,6	2,4	5,7,8,9,10
color r11	1,3	2,4,6	5,7,8,9,10
merge rf r11	1,4,6	2,3	5,7,8,9,10
merge mix r11	1,3	2,4,6	5,7,8,9,10
merge fr r11	1,3	2,4,6	5,7,8,9,10
parallel rf r11	1,4,6	2,3	5,7,8,9,10
parallel mix r11	1,4,6	2,3	5,7,8,9,10
parallel fr r11	1,4,6	2,3	5,7,8,9,10
change r11	1,4,6	2,3	5,7,8,9,10

表 A.3: Defferential Equation の演算器割り当て

	R1	R2	R3	R4	R5	R6	R7	R8
left r13	13,18	6,9,12, 14,16,19	7,20	5,15,21	11,22	3,17,23	4,24	8,10,25
left r14	13,18	6,9,12, 14,16,19	7,20	5,15,21	11,22	3,17,23	4,24	8,10,25
minTc r13	9,15,20	13,18	7,21	11,22	17,24	4	1,2,3,25	8,10,12,23
minTc r14	17,23	14,22	2,6	11,20	4,16	1,10,21	7,24	3,12,13,19
color r13	6,11,19	1,8,13,24	5,14,22	3,9,15,25	7,20	16,17	4,23	10,18
merge rf r16	1,21	2,23	3,12,13	4,24	5,18	6,14,19	7,20	8,11
merge mix r14	1,10,23	2,22	3,9,15,25	4,16	5,18	6,11	7,21	8,13,24
merge fr r16	1,2,3	4	5,6	8,11,19	7,24	9,18	10,15,23	17,25
parallel rf r13	1,2,3,8,23	9,18	13,22	4,25	5,15,16	6,10,14,19	7,20	11,12,21
parallel rf r14	9,18	12,19	7,20	15,5,21	14,10,22	17,23	4,24	8,11,25
parallel mix r14	13,18	19	7,20	1,2,21	10,22	4,23	5,8,15,24	3,17,25
parallel fr r14	13,18	12,19	9,15,20	7,21	11,22	3,17,23	8,10,14,24	4,25
change r14	10,18	8,12,19	7,20	9,15,21	5,14,22	3,17,23	4,24	11,25

	R9	R10	R11	R12	R13	R14	R15	R16
left r13	1,2	26	27	28	29			
left r14	1	2	26	27	28	29		
minTc r13	5,6,14, 16,19	26	27	28	29			
minTc r14	5,18	8,9,15,25	26	27	28	29		
color r13	2,12,21	26	27	28	29			
merge rf r16	9,15	10,25	22	16,17	26	27	28	29
merge mix r14	17,19	12,14,20	26	27	28	29		
merge fr r16	12,21	13,16	14,20	22	26	27	28	29
parallel rf r13	17,24	26	27	28	29			
parallel rf r14	1,2,6	3,13,16	26	27	28	29		
parallel mix r14	6,9,12,14	11,16	26	27	28	29		
parallel fr r14	5,6	1,2,16	26	27	28	29		
change r14	1,2,6	13,16	26	27	28	29		

表 A.4: Jaumann Wave Filter のレジスタ割り当て

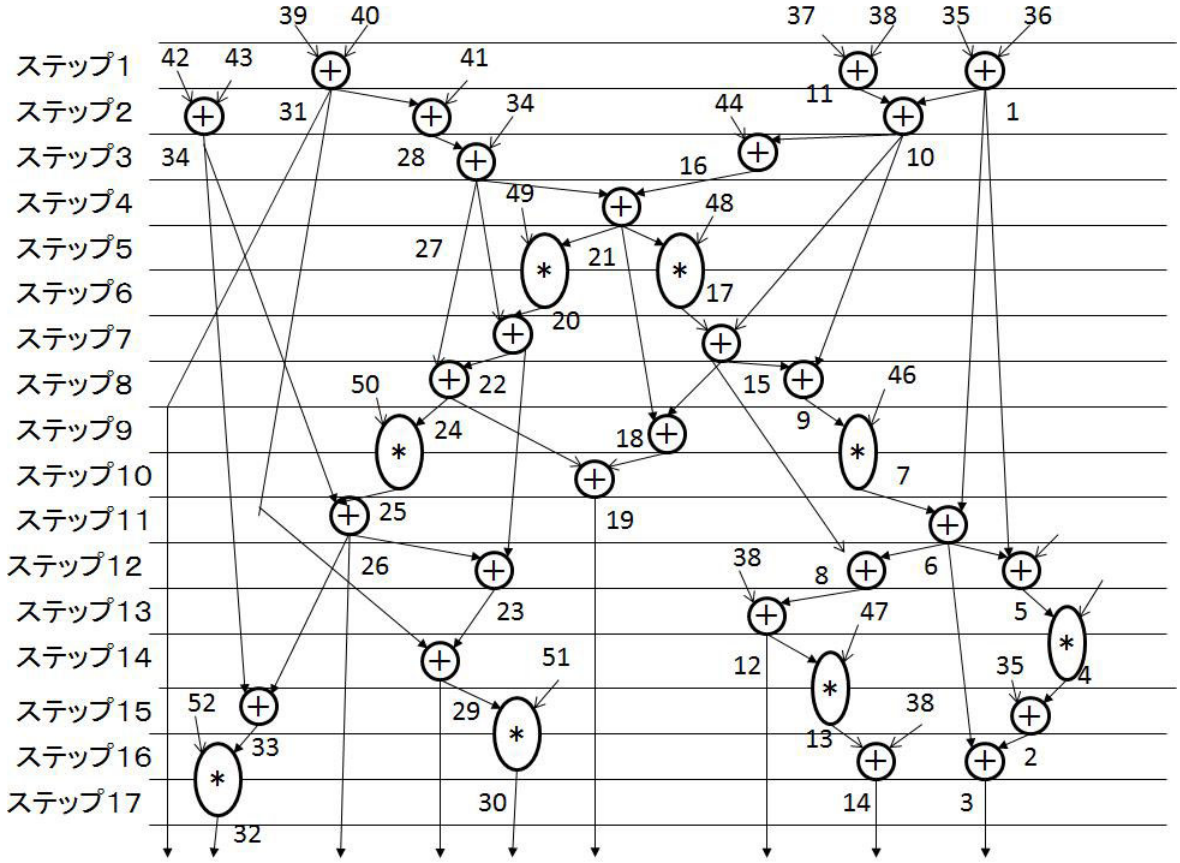


図 A.3: Elliptic Wave Filter

	F1(type1)	F2(type2)	F3(type2)
left r13	8,9,12,14	1,2,3,4,5,6,7,10,11,13,16	15,17
left r14	8,9,12,14	1,2,3,4,5,6,7,10,11,13,16	15,17
minTc r13	8,9,12,14	1,2,3,5,7,13,17	4,6,10,11,15,16
minTc r14	8,9,12,14	1,2,4,6,10,15	3,5,7,11,13,16,17
color r13	8,9,12,14	1,2,3,5,15,16,17	4,6,7,10,11,13
greedy rf r16	8,9,12,14	3,4,5,7,10,11,13,16	1,2,6,15,17
greedy mix r14	8,9,12,14	1,2,3,5,6,15,17	4,7,10,11,13,16
greedy fr r16	8,9,12,14	1,2,3,5,6,7,10,11,13,16,17	4,15
parallel rf r13	8,9,12,14	1,2,3,4,5,6,7,10,11,13,16	15,17
parallel rf r14	8,9,12,14	1,5,6,10,15,17	2,3,4,7,11,13,16
parallel mix r14	8,9,12,14	1,3,5,10,11,13,15,17	2,4,6,7,16
parallel fr r14	8,9,12,14	1,3,5,7,11,17	2,4,6,10,13,15,16
change r14	8,9,12,14	1,2,4,5,6,7,11,13,16	3,10,15,17

表 A.5: Jaumann Wave Filter の演算器割り当て

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
left r19	3,35,	1,2, 4,5, 14,36	6,7,9, 10,11, 30,37	32,38,	31,39,	16,18, 19,21, 28,40	13,34, 41	24,2,5 26,27, 42	8,12, 15,17, 43	20,22, 23,29, 44
left r20	3,35,	1,2, 4,5, 14,36	6,7,9, 10,11, 30,37	32,38,	31,39,	16,18, 19,21, 28,40	13,34, 41	24,2,5 26,27, 42	8,12, 15,17, 43	20,22, 23,44
minTc r19	4,11,13,16, 18,21,23, 28,30,40	2,8, 14,17, 22,42	25,26, 27,36	5,15, 20,29, 44	1,12	33,34, 41	38	31,39	9,19, 43	3,35
minTc r20	31	14,38	12,16, 22,28, 42	19,24, 27,44	33,34, 37	30,35	3,21, 23,43	4,7, 13,32 39	1,2, 5	9,10, 26,41
color r19	38	15,20, 23,33, 37	2,4,5, 11,16,22, 30,39	31,36	3,6, 18,21, 28,43	7,25, 44	13,24, 26,27, 34,41	1,8, 29,40	10,19, 32	9,12, 17,42
parallel rf r19	14,35	11,12, 21,25, 36	31,37	3,38,	4,5, 13,22, 32,39	6,17, 18,40	7,10, 24,26, 41	8,15,16, 20,21, 28,42	2,30, 34,43	9,19, 27,44
parallel rf r20	14,35	11,15, 20,33, 36	31,37	3,38	26,39	6,40	9,10, 19,41	8,16,17, 22,28, 29,42	2,32, 34,43	7,12, 24,27, 44
change r20	14,35	11,15, 20,33 36	31,37	3,38	7,18, 26,28, 39	6,16, 40	9,10, 19,32, 41	8,17, 22,29	2,34, 43	12,24, 27,44

	R11	R12	R13	R14	R15	R16	R17	R18	R19	R20
left r19	33	45	46	47	48	49	50	51	52	
left r20	29	33	45	46	47	48	49	50	51	52
minTc r19	6,7, 10,24, 32,37	45	46	47	48	49	50	51	52	
minTc r20	6,17, 18,25, 36	8,11, 15,20, 29,40	45	46	47	48	49	50	51	52
color r19	14,35	45	46	47	48	49	50	51	52	
parallel rf r19	1,23, 33	45	46	47	48	49	50	51	52	
parallel rf r20	1,4, 23	5,13, 18,21, 25,30	45	46	47	48	49	50	51	52
change rf r20	1,4, 23,30	5,13, 21,25	45	46	47	48	49	50	51	52

表 A.6: Elliptic Wave Filter のレジスタ割り当て

	F1(type1)	F2(type1)	F3(type2)	F4(type2)	F5(type2)
left r19	4,7,17,30	13,20,25,32	1,2,3,5,6,9,10, 12,15,16,18,19,21	8,11,14,22,24, 26,27,28,33	23,29,31,34
left r20	4,7,17,30	13,20,25,32	1,2,3,5,6,9,10, 12,15,16,18,19,21	8,11,14,22,24, 26,27,28,33	23,29,31,34
minTc r19	4,7,17,30	13,20,25,32	1,10,12,14,16,18,19,21, 22,23,24,26,29,33	2,6,8,9,11, 15,27,34	3,5,28,31
minTc r20	4,7,17,30	13,20,25,32	5,6,12,24,28,29,31,33	11,14,18,19, 22,23,27,34	1,2,3,8,9,10,15, 16,21,26
color r19	4,7,20,30	13,17,25,32	12,14,15,23,26,27,28,31, 33	2,3,5,6,11,16,18, 21,22,24,34	1,8,9,10,19,29
parallel rf r19	4,7,20,30	13,17,25,32	1,3,5,9,10,18,22,26, 27	6,8,11,12,14,15, 16,19,21,24,28,33	2,23,29,31,34
parallel rf r20	4,7,20,30	13,17,25,32	1,12,16,22,23,24,26,28, 29	8,11,14,33,34	2,3,5,6,9,10,15, 18.19,21,27,31
change r20	4,17,25,30	7,13,20,32	1,12,16,22,23,24,26,28, 29	8,11,14,33,34	2,3,5,6,9,10,15, 18.19,21,27,31

表 A.7: Elliptic Wave Filter の演算器割り当て