

Title	制御のタイミングスキューおよびストールに基づくLSIチューニング
Author(s)	上原, 八弓
Citation	
Issue Date	2009-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/8128
Rights	
Description	Supervisor:金子 峰雄, 情報科学研究科, 修士

修士論文

制御のタイミングスキューおよびストール
に基づくLSIチューニング

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

上原 八弓

2009年3月

修士論文

制御のタイミングスキューおよびストール に基づくLSIチューニング

指導教官 金子峰雄 教授

審査委員主査 金子峰雄 教授
審査委員 宮地充子 教授
審査委員 上原隆平 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システム学専攻

0710013 上原 八弓

提出年月: 2009年2月

概要

LSIの微細化に伴って、遅延量などのばらつきが相対的に大きくなってきており、最悪値評価に基づく設計では性能向上が難しくなっている。この問題に対して、製造後のLSIの一部チューニングによって性能を確保することが考えられる。本稿ではデータパス回路を対象として、制御タイミング・スキューと制御ストールによって性能劣化を最小限に止めて回路を正しく動作させる手法を提案する。特にここでは与えられたデータパス回路(構造記述と制御記述)と遅延情報とから、スキュー調整を許してストール数を最小化する問題を提起し、混合整数線形計画法に基づく解法を提案する。なお提案手法が最も効率的に機能するようなデータパス合成は今後の課題となっている。

目次

第1章	はじめに	1
第2章	高位合成	3
2.1	LSI設計の流れと高位合成	3
2.1.1	高位合成の概要	5
2.2	データフローグラフ	5
2.3	スケジューリング	6
2.4	資源割り当て	8
2.4.1	レジスタ割り当て	8
2.4.2	演算器割り当て	10
2.5	RTL回路	11
2.6	セットアップ・ホールド条件	11
第3章	本研究で提案する信号伝搬遅延のばらつきに適應する回路方式	14
3.1	タイミングスキュー	14
3.2	ストール	15
第4章	ストール数最小化問題の定式化	18
4.1	提案手法を用いた例	18
4.2	MILPによる問題の定式化	21
第5章	指定されたストール数以内でのストール数最小化問題の計算複雑度	22
5.1	スキュー制約グラフ	22
5.2	指定されたストール数以内でのストール数最小化問題の計算複雑度	25
第6章	評価実験	27
第7章	まとめと今後の課題	30

第1章 はじめに

近年，半導体製造技術の進歩に伴って，集積回路 (LSI:large scale integrated circuit) の微細化が進んでいる．これにより LSI の小型化や，過去のものと比較して同じ面積でより高機能な LSI の製造が可能となっている．しかしながらその一方で，製造のばらつきも拡大し，トランジスタの性能や配線抵抗などのばらつきにより信号伝播遅延のばらつきが遅延量に対し相対的に拡大し，深刻な問題ととなっている [1]．現在主流となっている同期式回路では，データを書き込むレジスタ間の信号伝播遅延の値をもとに，レジスタへデータを書き込むタイミングを決定している．従来の設計では，製造ばらつきによる信号伝播遅延のばらつきが遅延量に対し相対的に微小だったので，信号伝播遅延を見積もり，その最悪値にマージンを持たせていた．しかし，製造ばらつきの拡大により信号伝播遅延のばらつきが遅延量に対し相対的に拡大し，より高性能な LSI が要求される中で，従来の設計のように信号伝播遅延の最悪値にマージンを持たせ，所望の性能を確保することは困難となりつつある．また，レジスタへの書込みタイミングも信号伝播遅延のばらつきにより，設計したタイミングでレジスタにデータが書き込まれず，回路が正常に動作しなくなることを回避する必要がある．

このような遅延ばらつきの問題に対して，様々な対策が検討されている．レジスタへの書込みタイミングの信号伝播遅延のばらつきに対しては，レジスタへのクロック信号の到着時刻に相対的な順序制約を設けることで，回路の動作を保障する手法 [4] や，レジスタの書込みタイミングのずれであるスキューを補正するデスキューを行う手法 [6] などが提案されている．

またレジスタ間の信号伝播遅延に対しては，統計的な遅延解析を行い，製造後の遅延量を分析し，性能と歩留まりの兼ね合いを調整して設計する回路の性能を決定することで，従来の手法では信号伝播遅延の最悪値により所望の性能の確保が困難な問題に対処している [2][3]．しかしながら，従来の遅延量の最悪値にもとづく手法や統計的遅延解析を用いた手法でも，設計時に回路の性能を決定する必要があり，実際に製造されたチップが設計時に決定された性能よりも高い性能で動作する可能性がある場合に対処することはできない．

製造されたチップの性能に応じて回路を動作させる方法として，製造後に回路のクロック周期を調整することで対処することが考えられる．しかしながら，注目している回路ブロックと他の回路ブロックとのインターフェースを考えたとき，回路ブロック毎のクロック周期調整は，クロック信号の共通化の妨げとなるのみならず，回路ブロック間のスムーズなデータの授受の妨げともなることから，回路のクロック周期を変更せずに，遅延ばら

つきの問題に対処する必要がある。また、ある一部分の信号伝播遅延の値のみが悪化したとき、その箇所に対処するためだけに回路ブロック全体へのクロック周期の値を大きくする必要があり、回路全体の処理速度が低下する。

このような問題に対処するため、本研究では、クロック周期を変更することなく、各チップの遅延のばらつきの状態に応じて、製造後に一部の回路部品を調整することにより、回路を正常にかつ、高速に動作させる回路方式を提案し、最終的には回路設計手法の開発を目的としている。具体的にはレジスタ毎に書込みのタイミングの調整行うスキュー調整機構と、回路全体の書込みタイミングをクロック周期に同期して以降の書込みタイミングを遅らせるストール調整機構を設けることで、クロック周期を変更せずに遅延ばらつきに適応し、回路を正しくかつ、製造されたチップ毎に応じた性能で動作させる。

特に本稿ではレジスタ等への制御信号のタイミングスキューと制御コントロールステップのストールによって性能劣化を最小限に抑えて回路を正しく動作させる手法を提案する。ここではストール数を最小化問題を定式化し、混合整数線形計画法 (MILP) を用いた解法を提案する。

本稿は以下のように構成される。第2章では高位合成について説明する。第3章では本研究で提案する信号伝播遅延ばらつきに適応する回路方式であるスキュー調整機構とストール調整機構について説明する。第4章ではストール数最小化問題の定式化を、さらにストール数最小化問題を MILP で定式化を行う。第5章では指定されたストール数以内でのストール数最小化問題の計算複雑度について検討する。第6章では MILP を用いて評価実験を行い、その結果について考察する。第7章でまとめと今後の課題について述べる。

第2章 高位合成

2.1 LSI設計の流れと高位合成

ここではLSI設計の流れと高位合成について説明する。

図 2.1 に LSI 設計の流れを示す。LSI 設計は大きく 4 つの段階に分かれる。まず第 1 段階にシステム設計 (アルゴリズム設計, アーキテクチャ設計ともいう) を行い, 与えられたシステムの仕様から, ハードウェアで実現する部分とソフトウェアで実現する部分を切り分けを行い, ハードウェアにどのような動作を要求するかを決める。ハードウェアの動作するアルゴリズムを記述したものを動作記述と呼び, C 言語やアルゴリズムで行われる演算間の依存関係をグラフで表現したデータフローグラフなどで記述される。第 2 段階では, システム設計で得られた動作記述からレジスタ転送レベル回路を設計する機能設計を行う。レジスタ転送レベル回路では, データの演算を行う演算器や演算結果のデータを保持するレジスタ等から成る演算部 (データパス) とそれを制御する制御部 (コントローラ) から構成させる。コントローラは有限状態機械 (FMS:finite state machine) で記述される。第 3 段階では, レジスタ転送レベル回路から論理回路を設計する論理設計を行う。論理設計では, レジスタ転送レベル回路のデータパスやコントローラを AND や OR 等の論理素子やフリップフロップ等の記憶素子から成る論理回路に変換する。最後に第 4 段階では, LSI の製造プロセスに必要なマスクパターンを作成する。マスクパターンの作成では, 最初に構成要素の部品を LSI のどの位置に配置するかというフロアプランを決め, その後, 部品を構成する各ゲートの配置とその間の配線を決定する。

近年の半導体技術の進歩や, 電子機器の高機能化によって LSI の規模や複雑度が飛躍的に増大している。また携帯電話などの電子機器では, 製品のリリースサイクルが短くなりつつあり, LSI の設計の時間も短縮することが求められている。このため設計される LSI が急速に大規模化しているにもかかわらず設計期間が短くなっており, かつてのように人手による LSI の設計では対処することが困難になりつつある。このようなことから設計生産性を向上する必要がある。その方法のひとつとしてコンピュータ支援設計 (CAD:computer-aided design) あるいは設計自動化 (DA:design automation) がある。コンピュータを活用した設計自動化により, 設計の様々な工程を自動化し, 最適な設計を高速に求めることが可能となる。また, 人手で行う設計レベルをより抽象度の高い高位の設計レベルに引き上げることにより, より大規模なシステムを用意に人手で設計できるようになる。このようなことから, 近年 LSI 設計の自動化が進んでいる。機能設計の自動化を高位合成, 論理設計の自動化を論理合成, レイアウト設計の自動化を自動配置配線と呼び,

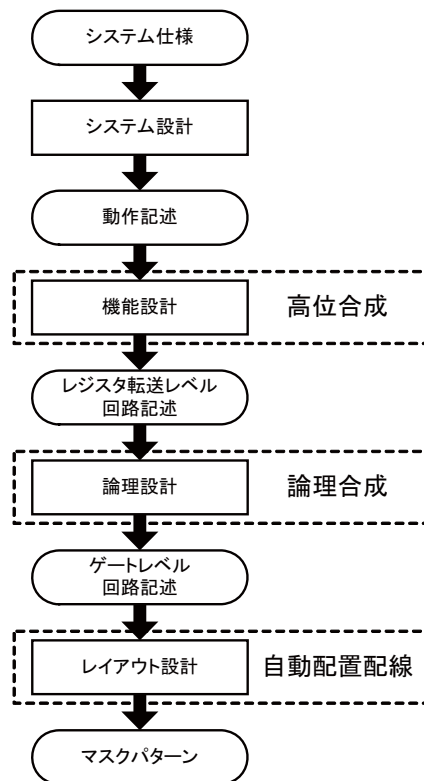


図 2.1: LSI 設計の流れ

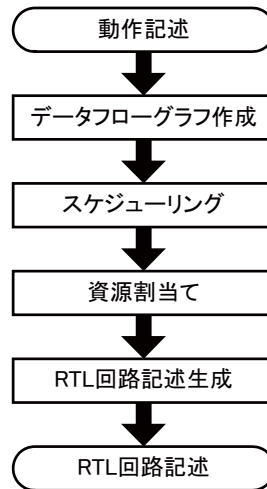


図 2.2: 高位合成の流れ

それぞれの分野で研究が行われている。

本研究では、新しいLSIの方式を提案し、将来的に高位合成でこの方式を活用した合成を行うことを想定している。

2.1.1 高位合成の概要

高位合成では回路の動作記述から、レジスタ転送レベル (RTL:register-transfer level) 回路変換する。代表的な高位合成の流れを図 2.2 に示す。ここでは回路の動作記述をデータフローグラフ (DFG:data-flow graph) で与える。高位合成ではデータフローグラフから、演算を行うタイミングを決定するスケジューリング、資源割り当てを行う。資源割り当ては、実際にどの演算器で演算を行うかを決定する演算器割り当て、演算結果のデータをどのレジスタで保持するかを決めるレジスタ割り当てから成る。そしてこれらの結果からレジスタ転送レベル回路を生成する。以降、これらについて説明する。

2.2 データフローグラフ

データフローグラフとはデータと演算の依存関係を示す有向グラフである。演算集合 O 、外部入力を表すダミー演算集合 P の和集合からなる頂点集合 $V = (O \cup P)$ 、演算間のデータの依存関係を有向辺 A としたときデータフローグラフは $G = (V, A)$ となる。図 2.13 にデータフローグラフの例を示す。図 2.13 では式 2.1 のようなデータと演算の依存関係を示している。

$$x = ((a + b) - c) + (d + e) \quad (2.1)$$

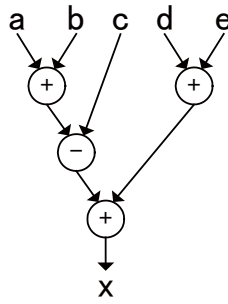


図 2.3: データフローグラフ

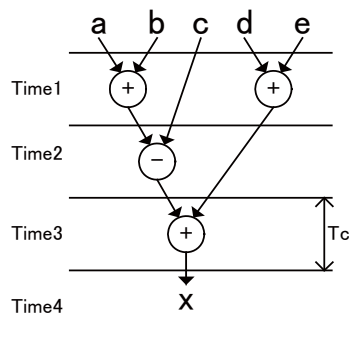


図 2.4: スケジューリング

図 2.13 のデータフローグラフではデータ a, b, c, d, e, x と演算操作 $+$ (加算), $-$ (減算) の依存関係を示している。

2.3 スケジューリング

スケジューリングではデータフローグラフで与えられた演算操作を、演算の順序制約を満たしながらどの時刻に実行するかを決め、スケジューリングは、演算集合 O を時刻を離散的に等間隔の時間 (クロック周期) に区切ったものであるコントロールステップを表す自然数 \mathbb{N} への写像 $\sigma : O \rightarrow \mathbb{N}$ である。図 2.13 のデータフローグラフをスケジューリングしたものを図 2.4 に示す。このスケジューリングでは時刻 0 ($Time0$) に演算 $a + b, d + e$ を、時刻 1 ($Time1$) に $a + b$ の結果と c の減算、時刻 2 ($Time2$) に $(a + b) - c$ と $d + e$ の加算を行っている。図 2.4 では同時に時刻 0 に加算の演算を 2 つ行っているため、加算の演算器が 2 つ必要になるが、図 2.5 のようなスケジューリングを行うと、同じ処理で加算器が 1 つで回路が構成できる。また図 2.4 や図 2.5 では各演算は 1 クロックサイクル内、つまりクロック周期 T_C 内で演算を終えているが、図 2.6 の乗算 ($*$) のような 2 サイクル演算などの多サイクルも存在する。

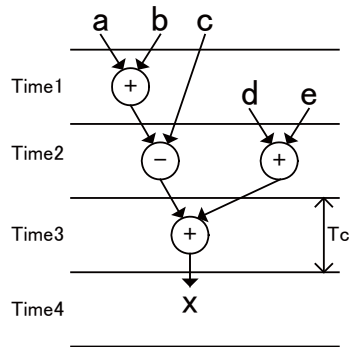


図 2.5: 1つの加算器でのスケジューリング



図 2.6: マルチサイクル演算のスケジューリング

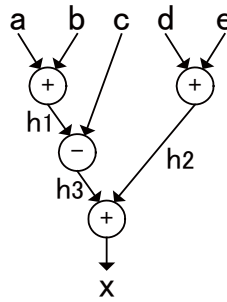


図 2.7: 内部変数を割り当てたデータフローグラフ

2.4 資源割り当て

データフローグラフから、データや演算結果を保持する内部変数をレジスタに割り当てる操作や、各演算を演算器に割り当てる操作を資源割り当てという。資源割り当てを行うことで、必要なレジスタ数や演算器数がわかり、またレジスタ数や演算器数に制約がある場合、これらの資源制約を考慮してスケジューリング、資源割り当てを行う。

2.4.1 レジスタ割り当て

レジスタ割り当ては、各変数データに対して、それを格納するレジスタを指定するもので、レジスタ集合を R 、データの集合を D としたとき、レジスタ割り当ては $\xi : D \rightarrow R$ となる。図 3.4 に内部変数を割り当てたデータフローグラフを示す。内部変数 h_1, h_2, h_3 は式 2.2, 2.3, 2.4 のようになる。

$$h_1 = a + b \quad (2.2)$$

$$h_2 = d + e \quad (2.3)$$

$$h_3 = h_1 - c \quad (2.4)$$

図 3.4 では $a + b$ の演算結果を $h_1, d + e$ を $h_2, a + b$ と c の減算結果を h_3 に割り当てている。これらの内部変数および入出力のデータは演算などに利用されるのを終えるまでその値を保持する。データに値が代入されてからその値が利用されるのを終えるまでの時間をライフタイムという。図 2.8 に図 2.4 のスケジューリングをもとに図 3.4 で割り当てた内部変数と入出力のデータのライフタイムを示す。次にこのようにして割り当てた内部変数や入出力のデータをレジスタに割り当てる。ライフタイムが重ならないデータはレジスタ共有が可能である。図 3.4 の内部変数を割り当てたデータフローグラフと図 2.4 のスケジューリングをもとにレジスタ割り当てを行ったものを図 2.9 に示す。図 2.9 では R_1, R_2, R_3, R_4, R_5 の 5 つのレジスタに各データと内部変数を割り当てている。レジスタと割り当てられている各データ、内部変数の関係を表 2.1 に示す。

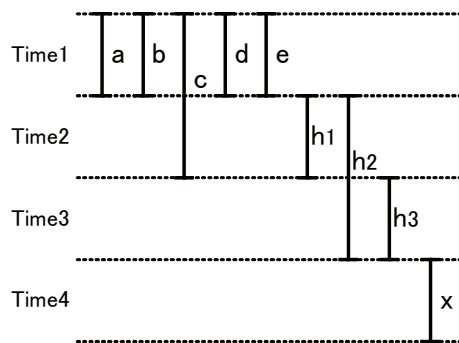


図 2.8: 内部変数と入出力データのライフタイム

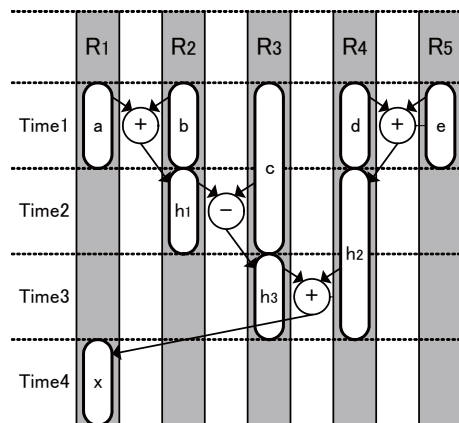


図 2.9: レジスタ割り当て

表 2.1: レジスタと各データ, 内部変数の割り当て

register	data and internal variable
R_1	a, x
R_2	b, h_1
R_3	c, h_3
R_4	d, h_2
R_5	c

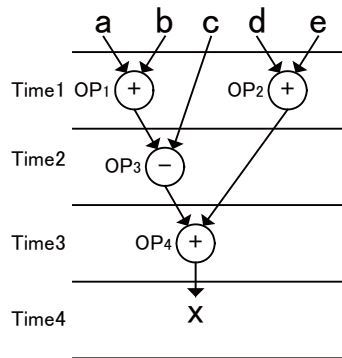


図 2.10: 演算器に名称を付けたデータフローグラフ

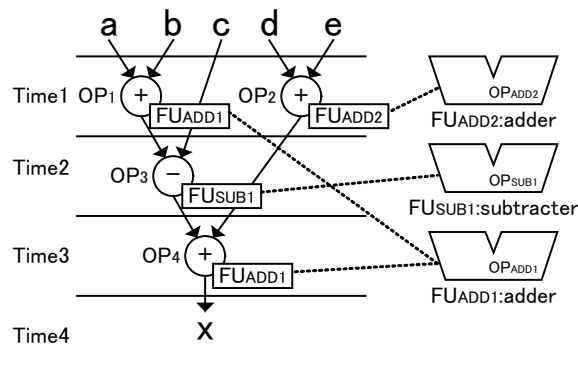


図 2.11: 演算器割り当て

2.4.2 演算器割り当て

演算器割り当ては、データフローグラフにおいて各演算に演算器を割り当てることである。演算集合 O 、演算器集合 F としたとき、演算器割り当ては $\rho: O \rightarrow F$ となる。説明の便宜上、図 2.4 のスケジューリングされたデータフローグラフの各演算に名前を付けたものを図 2.10 に示す。図 2.10 では a と b の加算の演算を OP_1 、 d と e の加算の演算を OP_2 、 OP_1 の演算結果と c の加算の演算を OP_3 、 OP_3 の演算結果と OP_2 の演算を OP_4 としている。図 2.11 に図 2.4 のスケジューリングされたデータフローグラフに演算器割り当てを行ったものを示す。ここでは加算器 (adder: FU_{ADD1}, FU_{ADD2}) と減算器 (subtractor: FU_{SUB1}) の合計 3 つの演算器を図 2.4 の各演算に割り当てている。図 2.4 のスケジューリングでは時刻 $Time1$ に加算の演算を同時に 2 つ行っているので FU_{ADD1}, FU_{ADD2} の 2 つの加算器が必要となるが、図 2.5 のようにスケジューリングを工夫することによって同時刻に行う演算を減らし、演算器の数を削減し、加算器を FU_{ADD1} のみにすることもできる。表 2.2 に図 2.11 の演算器割り当ての各演算の演算器の割り当て結果を示す。

表 2.2 では演算器 OP_{ADD1} に演算 OP_1, OP_4 、演算器 OP_{ADD2} に演算 OP_2 、演算器 OP_{SUB1}

表 2.2: 各演算と演算器の割り当て

functional unit	operation
FU_{ADD1}	OP_1, OP_4
FU_{ADD2}	OP_2
FU_{SUB1}	OP_3

に演算 OP_3 を割り当てていることを示している。

レジスタ割り当てや演算器割り当てでは、その方法によってレジスタ数や演算器の数などの資源数が異なる。一般的にレジスタや演算器はできるだけ少ない数で回路を構成するように資源割り当てを行う。これまで説明したとおり資源割り当てはスケジューリングと密接に関係している。スケジューリングでは一般的にデータフローグラフなどで与えられた演算や動作をなるべく少ない実行ステップ数で行うようにスケジューリングする。しかしレジスタ数を最小化したときスケジューリングの実行ステップ数の値が大きくなってしまったり、逆にスケジューリングの実行ステップ数を最小化したとき、レジスタ数や演算器数の値が大きくなることもあり、スケジューリングと資源割り当ての関係がトレードオフになる場合がある。

2.5 RTL 回路

スケジューリングおよび資源割り当ての結果から RTL 回路を生成することができる。RTL 回路はデータパスとコントローラから構成されている。図 2.12 に図 2.13 のデータフローグラフから図 2.4 のスケジューリング、表 2.1, 2.2 の資源割り当てを行って生成した RTL 回路を示す。データパスではレジスタや演算器などの接続を示し、コントローラでマルチプレクサ (複数の入力から一つを選択し出力する回路) の信号選択制御やレジスタのデータ書き込みタイミング信号を制御する。

2.6 セットアップ・ホールド条件

はじめに、レジスタに正しく演算結果が書き込まれる条件であるセットアップ条件とホールド条件について説明する。

図 2.13 に、実装対象となる演算と演算間のデータの依存関係を表わしたデータフローグラフに、各演算をどの時刻に行うかを定めるスケジューリング、各演算を行う演算器を決める演算器割り当て、各演算の演算結果のデータを書き込むレジスタを決めるレジスタ割り当てを行った一例を示す。すなわち、演算 o_2 は、演算 o_1 の演算結果を入力として、演算器 f_1 の上で実行される。また、演算 o_1, o_3 の演算結果をレジスタ r_1 に、演算 o_2 の演

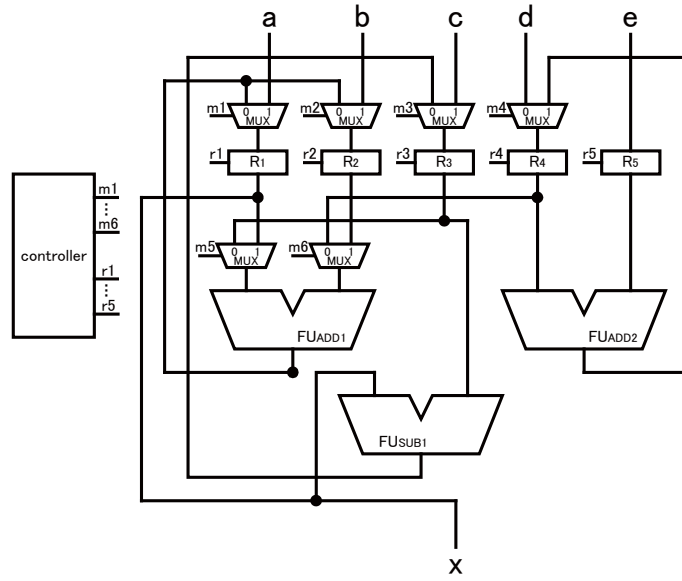


図 2.12: RTL 回路

算結果をレジスタ r_2 に書き込んでいる．これらのタイミング図を図 2.14 に示す．演算 o_1 , o_2 , o_3 の演算結果のデータをレジスタに書込むクロックに同期したタイミングを $\sigma(o_1)$, $\sigma(o_2)$, $\sigma(o_3)$, クロック周期を T_c , レジスタ r_1 から演算器 f_1 を経てレジスタ r_2 へ信号が伝搬する最大遅延を $D_{r_1 \rightarrow f_1 \rightarrow r_2}^{max}$, 最小遅延を $D_{r_1 \rightarrow f_1 \rightarrow r_2}^{min}$ とする．演算 o_2 の演算結果をレジスタ r_2 へ書き込むには，演算 o_1 の結果を書き込んだレジスタ r_1 から信号が出力され，演算器 f_1 を経て，レジスタ r_2 に信号 (o_2 の演算結果) が到達した後に r_2 にデータを書き込まなければならない．この条件をセットアップ条件といい，式 (2.5) にて表わされる．

$$\sigma(o_1) \cdot T_c + D_{r_1 \rightarrow f_1 \rightarrow r_2}^{max} \leq \sigma(o_2) \cdot T_c \quad (2.5)$$

一方，レジスタ r_1 にある演算 o_1 の演算結果は演算 o_3 の結果によって上書きされる．この状況において，演算 o_2 の結果がレジスタ r_2 に正しく書き込まれるためには，レジスタ r_1 から出力される o_3 の結果の信号が，演算器 f_1 を経てレジスタ r_2 に到達する以前に演算 o_2 の結果をレジスタ r_2 に書き込まなければならない．この条件をホールド条件といい，式 (2.6) で表わされる．

$$\sigma(o_3) \cdot T_c + D_{r_1 \rightarrow f_1 \rightarrow r_2}^{min} > \sigma(o_2) \cdot T_c \quad (2.6)$$

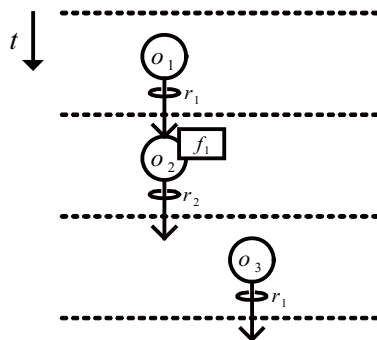


図 2.13: データフローグラフと資源割り当て

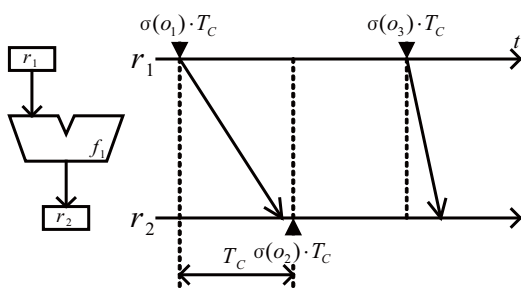


図 2.14: セットアップ条件とホールド条件

第3章 本研究で提案する信号伝搬遅延のばらつきに適応する回路方式

LSIの製造ばらつきにより、信号伝搬遅延がばらつく。この為、各レジスタ間の最大遅延、最小遅延がばらつき、設計時に見積もった値を外れ、レジスタに正しいデータが書き込まれる条件であるセットアップ・ホールド条件を満たさなくなる可能性がある。本研究では、製造ばらつきによる信号伝搬遅延のばらつきに適応する為、以下に説明するタイミングスキュー調整とストール操作を製造後に行い、すべての演算がセットアップ・ホールド条件を満たすようにする。

3.1 タイミングスキュー

レジスタの書き込み制御信号の到着時刻は、クロックと同期している。このレジスタの書き込み制御信号の到着時刻にレジスタ毎のズレを持たせることにより、信号伝搬遅延ばらつきによって生じるセットアップ・ホールド条件違反を回避することが考えられる。本稿ではレジスタごとに独立にスキューの値を設定することができるものとし、レジスタ r_i のスキューの値を $\tau(r_i)$ とする。

図3.1は、製造ばらつきにより、レジスタ r_1 から演算器 f_1 を経て、レジスタ r_2 へ到達するまでの最大遅延の値が大きくなり、 r_1 を入力レジスタ、 r_2 を出力レジスタとする o_2 の演算に対するセットアップ条件が満たされなくなった場合に、レジスタ r_2 の書き込み制御信号のタイミングを $\tau(r_2)$ だけ遅らせることにより、セットアップ条件を満たすことができる例を示している。

図3.2はスキューを考慮したタイミング図を表しており、スキューを考慮したセットアップ条件式は式(3.1)、ホールド条件式は式(3.2)のように表わされる。

$$\begin{aligned} \sigma(o_1) \cdot T_c + \tau(r_1) + D_{r_1 \rightarrow f_1 \rightarrow r_2}^{max} \\ + \leq \sigma(o_2) \cdot T_c + \tau(r_2) \end{aligned} \quad (3.1)$$

$$\begin{aligned} \sigma(o_3) \cdot T_c + \tau(r_1) + D_{r_1 \rightarrow f_1 \rightarrow r_2}^{min} \\ + > \sigma(o_2) \cdot T_c + \tau(r_2) \end{aligned} \quad (3.2)$$

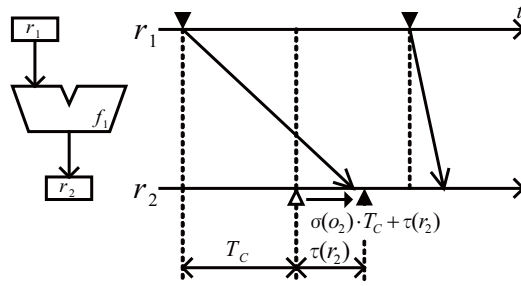


図 3.1: スキューを用いたセットアップ違反回避

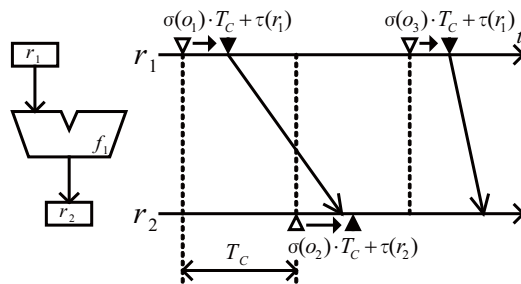


図 3.2: スキューを考慮したセットアップ・ホールド条件

3.2 ストール

ここで考えるストールは、本来刻まれるべき制御ステップの刻みを見送ることで、以降のタイミングを時間軸方向で平行移動する操作である。図 3.3 は、レジスタ r_1 のデータを入力とし、演算器 f_1 にて実行される演算 o_2 の結果のレジスタ r_2 への書き込みについてのセットアップ条件違反を、ストールにより回避する様子を示している。また、これに対応するスケジュール図を図 3.4 に示す。

ストールされたコントロールステップ ($\sigma(o_1)$) 以降の演算は、すべてストールしたステップ数だけ後にスケジュールが変更されることになる。本研究で提案する回路方式では、データパス制御部に、あらかじめストールを行うための機構を設けておき、製造後に各コントロールステップのストール数を調整できるようにするものとする。コントロールステップ i のストール数を $f_s(i)$ とするとき、ストールを考慮したセットアップ・ホールド条件式は

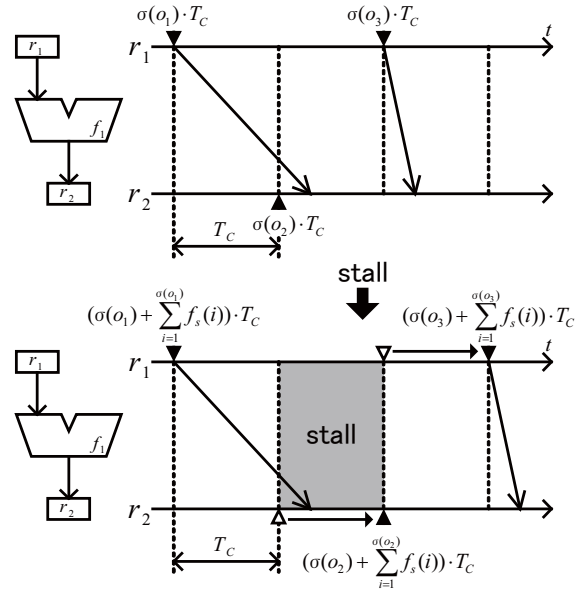


図 3.3: ストール操作によるセットアップ条件違反の回避

以下のようになる .

$$\begin{aligned}
 (\sigma(o_1) + \sum_{i=1}^{\sigma(o_1)} f_s(i)) \cdot T_c + D_{r_1 \rightarrow f_1 \rightarrow r_2}^{max} \\
 \leq (\sigma(o_2) + \sum_{i=1}^{\sigma(o_2)} f_s(i)) \cdot T_c
 \end{aligned} \tag{3.3}$$

$$\begin{aligned}
 (\sigma(o_3) + \sum_{i=1}^{\sigma(o_3)} f_s(i)) \cdot T_c + D_{r_1 \rightarrow f_1 \rightarrow r_2}^{min} \\
 > (\sigma(o_2) + \sum_{i=1}^{\sigma(o_2)} f_s(i)) \cdot T_c
 \end{aligned} \tag{3.4}$$

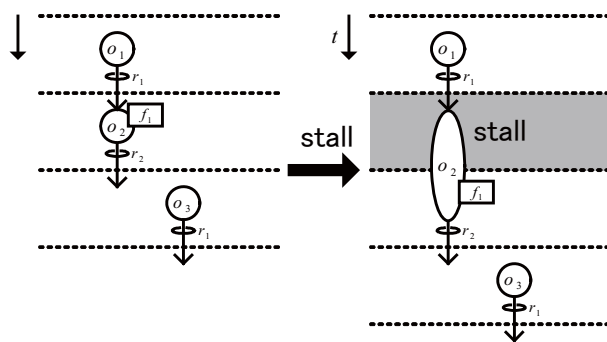


図 3.4: スケジュール上でのストール操作

第4章 ストール数最小化問題の定式化

ここでは、製造後に各レジスタのスキューの値と各コントロールステップのストール数を調整できる回路方式のもとで、製造後に回路全体のストール数を最小化する問題を考える。

実装する計算が、演算の集合を O 、演算間のデータの依存関係を有向辺の集合 D としたデータフローグラフ $G = (O, D)$ にて与えられ、更に、クロックに同期した演算結果の書き込みタイミングのスケジュール $\sigma : O \rightarrow \mathbb{N}$ 、演算器の集合 F 、レジスタの集合 R 、演算器割り当て $\rho : O \rightarrow F$ 、演算結果のデータを書き込むレジスタを特定するためのレジスタ割り当て $\xi : O \rightarrow R$ 、レジスタ r_i から演算器 f_x を経てレジスタ r_j へデータの信号が到達する最大遅延 $D_{r_i \rightarrow f_x \rightarrow r_j}^{max} \in \mathbb{R}$ 、最小遅延 $D_{r_i \rightarrow f_x \rightarrow r_j}^{min} \in \mathbb{R}$ 、クロック周期 $T_c \in \mathbb{R}$ が与えられ、レジスタ r_i のスキューの値 $\tau(r_i) \in \mathbb{R}$ と、コントロールステップ $i \in \mathbb{N}$ のストール数 $f_s(i) \in \mathbb{N}$ を求める問題とする。

[入力]	データフローグラフ (G)、スケジュール (σ)、 演算器割り当て (ρ)、レジスタ割り当て (ξ)、 遅延情報 (D^{max}, D^{min})、 クロック周期 T_c
[出力]	各レジスタのスキューの値 (τ)、 各コントロールステップのストール数 (f_s)
[制約条件]	全ての演算に対するセットアップ条件、 ホールド条件
[最適化目標]	ストール数最小化

4.1 提案手法を用いた例

提案方式の有用性を、図 4.1 に示すスケジューリングと資源割り当てが行われたデータフローグラフを例に説明する。図 4.1 では、演算 o_2 を演算器 f_1 の上で実行し、 o_2 の演算結果をレジスタ r_1 に書き込み、次にレジスタ r_2 に書き込まれた演算 o_2 の結果を用いて演算 o_3 を演算器 f_2 の上で実行して、その o_3 の演算結果をレジスタ r_1 に書き込んでいる。このとき、レジスタ r_1 から演算器 f_1 を経てレジスタ r_2 へ到達する最大遅延 $D_{r_1 \rightarrow f_1 \rightarrow r_2}^{max}$ と

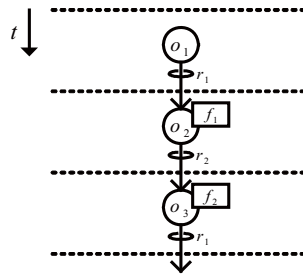


図 4.1: スケジューリング・資源割り当て済みのデータフローグラフ

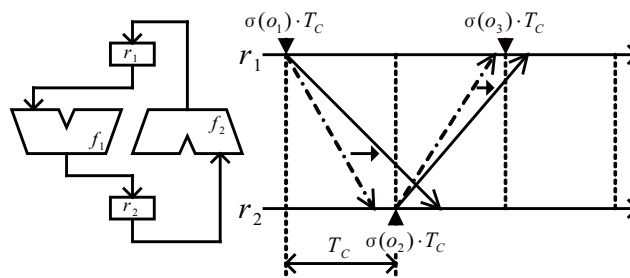


図 4.2: 従来方式の場合

レジスタ r_2 から演算器 f_2 を経てレジスタ r_1 へ到達する最大遅延 $D_{r_2 \rightarrow f_2 \rightarrow r_1}^{max}$ の設計時見積もりは、共にクロック周期 T_c 未満とする。

製造ばらつきにより、 $D_{r_1 \rightarrow f_1 \rightarrow r_2}^{max}$ と、 $D_{r_2 \rightarrow f_2 \rightarrow r_1}^{max}$ の値がそれぞれ増加し、演算 o_2 、 o_3 の演算結果を書き込むタイミングよりも演算結果の到着が遅くなったとき、スキュー・ストールがない従来の回路方式において、それぞれの演算においてセットアップ条件を満たすことができず、レジスタ r_1 、レジスタ r_2 に正しい演算結果が書きこまれない。また、各レジスタのスキューの値のみ調整可能な場合では、スキューは各レジスタごとに固有の値をとるため、一方の演算（たとえば、 r_1 から r_2 への遅延によって支配される演算 o_2 ）にあわせてスキューを調整してセットアップ条件を満たせたとしても、他方の演算（ r_2 から r_1 への遅延によって支配される演算 o_3 ）については、セットアップ条件違反を回避できない。

ストール調整のみが可能な場合では、図 4.4 のようにストールを行うことで、セットアップ条件を満たし、正しい演算結果がレジスタに書き込まれるが、全体のコントロールステップ数は 2 ステップ増加する。

これらに対し提案手法では、スキューとストールを共に調整することによって、セットアップ条件を満たし、全体のコントロールステップ数の増加を 1 ステップにとどめることができ、ストール調整のみの構成と比べて、コントロールステップ数を 1 ステップ削減することができる。

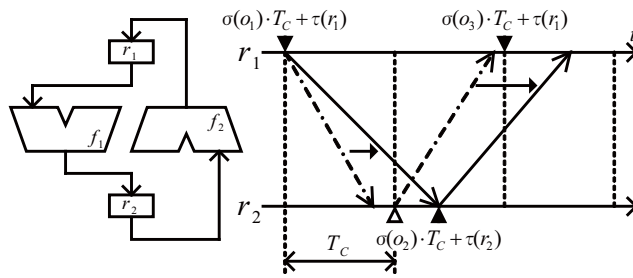


図 4.3: スキューのみの場合

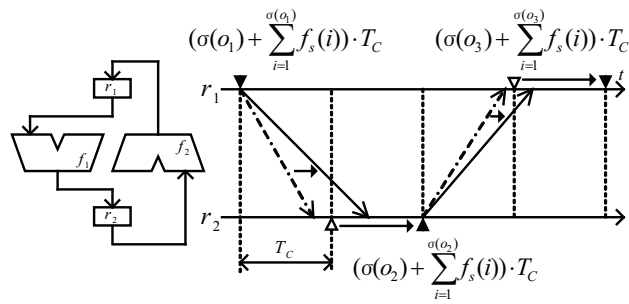


図 4.4: ストールをみの場合

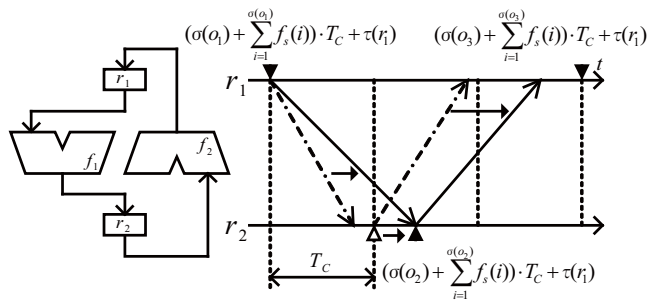


図 4.5: スキュー・ストールを用いた場合

4.2 MILP による問題の定式化

ここでは、スキューとストール調整の下でのストール数最小化問題を混合整数線形計画法にて解く為の定式化を行う。

クロックに同期した演算 o_i の書き込みタイミングを $\sigma(o_i) \in \mathbb{N}$ 、レジスタ r_i のスキューの値 $\tau(r_i) \in \mathbb{R}$ 、コントロールステップ $i \in \mathbb{N}$ のストール数を $f_s(i) \in \mathbb{N}$ 、クロック周期を $T_c \in \mathbb{R}$ 、レジスタ r_i から演算器 f_x を経てレジスタ r_j に信号が到達する最大遅延を $D_{r_i \rightarrow f_x \rightarrow r_j}^{max} \in \mathbb{R}$ 、最小遅延を $D_{r_i \rightarrow f_x \rightarrow r_j}^{min} \in \mathbb{R}$ としたとき、 $\sigma(o_i)$ 、 T_c 、 D^{max} 、 D^{min} が定数変数となり、各レジスタのスキューの値 $\tau(r_i)$ と、各コントロールステップのストール数 $f_s(i)$ が未知変数となる。これら変数を使って、セットアップ・ホールド条件は以下のように書き表わされる。

$$\begin{aligned} (\sigma(o_1) + \sum_{i=1}^{\sigma(o_1)} f_s(i)) \cdot T_c + \tau(r_1) + D_{r_1 \rightarrow f_1 \rightarrow r_2}^{max} \\ \leq (\sigma(o_2) + \sum_{i=1}^{\sigma(o_2)} f_s(i)) \cdot T_c + \tau(r_2) \end{aligned} \quad (4.1)$$

$$\begin{aligned} (\sigma(o_3) + \sum_{i=1}^{\sigma(o_3)} f_s(i)) \cdot T_c + \tau(r_1) + D_{r_1 \rightarrow f_1 \rightarrow r_2}^{min} \\ > (\sigma(o_2) + \sum_{i=1}^{\sigma(o_2)} f_s(i)) \cdot T_c + \tau(r_2) \end{aligned} \quad (4.2)$$

最適化目標である回路全体のストール数最小化は以下のようなになる。

$$\sum_{i=0}^{\max_j(\sigma(o_j))} f_s(i) \rightarrow \min \quad (4.3)$$

第5章 指定されたストール数以内でのストール数最小化問題の計算複雑度

ここでは指定されたストール数以内で，ストール数を最小化する問題の計算複雑度について述べる．

5.1 スキュー制約グラフ

ここでは，スキュー制約グラフについて説明する．スキュー制約グラフは，レジスタ r_i のスキューの値 $\tau(r_i)$ を頂点とする有向グラフである． r_1 のスキューの値を $\tau(r_1)$ ， r_2 のスキューの値を $\tau(r_2)$ ， f_1 の最大遅延の値を $D_{r_1 \rightarrow f_1 \rightarrow r_2}^{max}$ ，最小遅延の値を $D_{r_1 \rightarrow f_1 \rightarrow r_2}^{min}$ とする． r_1 のデータ書き込みタイミングを $\sigma(o_1)$ ， r_2 の書き込みタイミングを $\sigma(o_2)$ とする．また， r_1 は， $\sigma(o_1)$ のデータの書き込みタイミング後にデータの上書きがあり，そのタイミングを $\sigma(o_3)$ と表す．このときのセットアップ・ホールド条件は以下ようになる．

$$\begin{aligned} \sigma(o_1) \cdot T_C + \tau(r_1) + D_{r_1 \rightarrow f_1 \rightarrow r_2}^{max} \\ \leq \sigma(o_2) \cdot T_2 + \tau(r_2) \end{aligned} \quad (5.1)$$

$$\begin{aligned} \sigma(o_3) \cdot T_C + \tau(r_1) + D_{r_1 \rightarrow f_1 \rightarrow r_2}^{min} \\ > \sigma(o_2) \cdot T_2 + \tau(r_2) \end{aligned} \quad (5.2)$$

これらの式を以下のように変形する．

$$\begin{aligned} \tau(r_2) - \tau(r_1) \\ \geq (\sigma(o_1) - \sigma(o_2)) \cdot T_C + D_{r_1 \rightarrow f_1 \rightarrow r_2}^{max} \end{aligned} \quad (5.3)$$

$$\begin{aligned} \tau(r_1) - \tau(r_2) \\ > (\sigma(o_2) - \sigma(o_3)) \cdot T_C - D_{r_1 \rightarrow f_1 \rightarrow r_2}^{min} \end{aligned} \quad (5.4)$$

このとき，スキュー制約グラフでは $\tau(r_1)$ ， $\tau(r_2)$ を頂点とし，頂点 $\tau(r_1)$ から $\tau(r_2)$ へ，重み $(\sigma(o_1) - \sigma(o_2)) \cdot T_C + D_{r_1 \rightarrow f_1 \rightarrow r_2}^{max}$ の有向辺で結ばれる．また，頂点 $\tau(r_2)$ から r_1 へ，重み $(\sigma(o_2) - \sigma(o_3)) \cdot T_C - D_{r_1 \rightarrow f_1 \rightarrow r_2}^{min}$ の有向辺で結ばれる．これらをグラフに表すと図 5.1 のようになる．

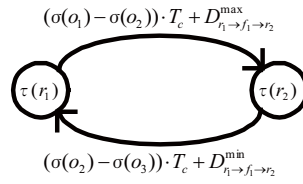


図 5.1: スキュー制約グラフ

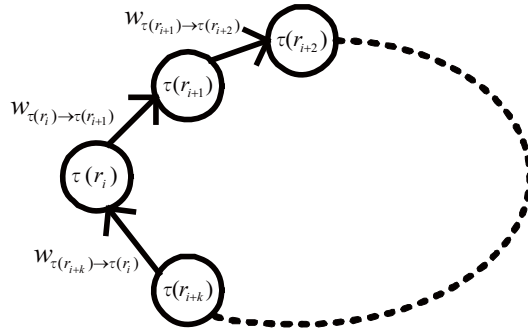


図 5.2: 正サイクルのスキュー制約グラフ

定理 5.1.1 すべてのスキューの値がセットアップ・ホールド条件を満たすように調節することが可能な必要十分条件は，スキュー制約グラフにおいて正サイクルがないことである．

証明 1 まず，すべてのスキューの値がセットアップ・ホールド条件を満たすように調節することが可能ならば，スキュー制約グラフにおいて正サイクルがないことについて，背理法を用いて証明する．すべてのスキューの値がセットアップ・ホールド条件を満たすように調節することが可能，かつスキュー制約グラフにおいて正サイクルがあると仮定する．このとき，図 5.2 正サイクル c_p に含まれる頂点 $\tau(r_i)$ について考えたとき，頂点 $\tau(r_x)$ から頂点 r_y への辺の重みを $w_{\tau(r_x) \rightarrow \tau(r_y)}$ ，正サイクルの辺の重みの和 $w_{sum} = w_{\tau(r_i) \rightarrow \tau(r_{i+1})} + w_{\tau(r_{i+1}) \rightarrow \tau(r_{i+2})} + \dots + w_{\tau(r_{i+k}) \rightarrow \tau(r_i)}$ とする．このとき，正サイクル c_p に含まれる頂点 $\tau(r_i), \tau(r_{i+1}), \tau(r_{i+2}), \dots, \tau(r_{i+k})$ は以下のように表わされる．

$$\begin{aligned}
 \tau(r_i) &\geq \tau(r_{i+1}) + w_{\tau(r_i) \rightarrow \tau(r_{i+1})} \\
 \tau(r_{i+1}) &\geq \tau(r_{i+2}) + w_{\tau(r_{i+1}) \rightarrow \tau(r_{i+2})} \\
 &\vdots \\
 \tau(r_{i+k}) &\geq \tau(r_i) + w_{\tau(r_{i+k}) \rightarrow \tau(r_i)}
 \end{aligned}$$

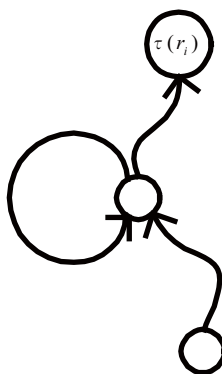


図 5.3: 頂点 $\tau(r_i)$ へのウォーク

これらの式の辺々を足し合わせると以下ようになる .

$$\begin{aligned}
 0 &\geq w_{\tau(r_{r+1}) \rightarrow \tau(r_{i+2})} + \cdots + w_{\tau(r_{i+k}) \rightarrow \tau(r_i)} \\
 0 &\geq w_{sum}
 \end{aligned}$$

$0 \geq w_{sum}$ より, c_p が正サイクルであることに矛盾する . よって, すべてのスキューの値がセットアップ・ホールド条件を満たすように調節することが可能, かつスキュー制約グラフにおいて正サイクルがある仮定に誤りがる . よって, すべてのスキューの値がセットアップ・ホールド条件を満たすように調節することが可能ならば, スキュー制約グラフにおいて正サイクルがない . 次に, スキュー制約グラフにおいて正サイクルがないならばすべてのスキューの値がセットアップ・ホールド条件を満たすように調節することが可能について証明する . 頂点 $\tau(r_i)$ のスキューの値は, 頂点 $\tau(r_i)$ へのウォークの辺の重み和の最大値以上に設定すれば, セットアップ・ホールド条件を満たす . 図 5.3 のような, ある頂点 $\tau(r_i)$ へのウォークを考えたとき, ウォークにサイクルが含まれていても, サイクルは正サイクルではないので, 頂点 $\tau(r_i)$ へ最大の辺の重み和を持つようなウォークから, サイクルを除去しても, 辺の重み和は同じか増えるので, 辺の重みの和が最大のウォークはパスとしてもよい . したがって, 頂点 $\tau(r_i)$ への辺の重みの和が最大のウォークはパスとして考えることができ, $\tau(r_i)$ のスキューの値を, このパスの辺の重み和より大きい値に設定すれば, セットアップ・ホールド条件を満たす . よって, スキュー制約グラフにおいて正サイクルがないならばすべてのスキューの値がセットアップ・ホールド条件を満たすように調節することが可能である . したがって, すべてのスキューの値がセットアップ・ホールド条件を満たすように調節することが可能な必要十分条件は, スキュー制約グラフにおいて正サイクルがないことである .

5.2 指定されたストール数以内でのストール数最小化問題の計算複雑度

元の総ステップ数 n , ストール数 i としたとき , ストール数 i を $n-1$ 個ステップで区切り , 並んだ結果だけを見て , 左から順にステップ 1 と 2 の区切り , ステップ 2 と 3 の区切り , \dots , ステップ $n-1$ とステップ n の区切りと解釈する . このときすべてのストールの組み合わせは , i 個のストールと $n-1$ 個の区切り計 $i+n-1$ 個の要素の並びは $(i+n-1)!$ となる . 但し , i 個のストールの並び順 , $n-1$ 個の区切りの並び順は区別しない . したがって , 元の総ステップ数 n , ストール数 i としたときのすべてのストールの組み合わせは ,

$$\frac{(i+n-1)!}{i! \cdot (n-1)!} \text{通り} \quad (5.5)$$

となる .

このことから最大ストール数を s としたとき , ストール数が $0 \sim s$ のすべてのストールの組み合わせは

$$\sum_{i=0}^s \frac{(i+n-1)!}{i! \cdot (n-1)!} \text{通り} \quad (5.6)$$

となる .

各ストールの組合せは , スキュー制約グラフを作成し , ベルマン・フォード法を用いることでスキュー制約グラフに正サイクルがないかを判定する . ベルマン・フォード法は以下のようになる .

入力: スキュー制約グラフ $G = (V, A)$

出力: 正サイクルがあれば TRUE を , そうでなければ FALSE を返す .

- 頂点 v_i までのパス長: $p(v_i)$,
- 辺 (v_i, v_j) の辺の重み: $w(v_i, v_j)$

Step0 $v_1 \cdots v_n$ の先行頂点となるソース頂点 v_{src} を付加し , そのソース頂点 v_{src} と $v_1 \cdots v_n$ を結ぶ有向辺の重みを $w(v_{src}, v_i) = 0$ とする .

Step1 すべての頂点 $v_i \in V$ に対して , $p(v_i) \leftarrow w(v_{src}, v_i)$ また , $k = 1$ とする .

Step2 すべての辺 $(v_i, v_j) \in A$ に対して , $p(v_j) < p(v_i) + w(v_i, v_j)$ ならば $p(v_j) \leftarrow p(v_i) + w(v_i, v_j)$

Step3 1. Step2 で p の更新が全くなされなければ停止し FALSE を返す .
2. p が更新されたとき ,

(a) $k < |V| + 1$ ならば $k \leftarrow k + 1$ として Step2 へ戻る .

(b) $k = |V| + 1$ ならば停止し , TRUE を返す .

スキュー制約グラフの頂点数を n , 辺の数を e としたとき , 全ての辺についての処理を最大頂点数+1 回繰り返すので , ベルマン・フォード法の計算量は $O((n + 1) \cdot e)$ となる .
すべてのストールの組み合わせに対して , 正サイクルの有無を判定する計算量は

$$O\left(\sum_{i=0}^s \frac{(i + n - 1)!}{i! \cdot (n - 1)!} ((n + 1) \cdot e)\right) \quad (5.7)$$

であることから , s が定数であるとき , ストール数最小化問題は多項式時間アルゴリズムを持つ .

第6章 評価実験

提案手法の有用性を評価するために、いくつかのベンチマークに対し、提案手法とクロック周期のみを調整する手法、ストールのみを調整する手法との比較を行った。MILPの解法には、CPLEXバージョン11.0.0を用い、プロセッサ:2.4GHz AMD(R)Dual Opteron、メモリ8GB RAMの上で実行した。ベンチマークとして、fourth-order Jaumann wave digital filter(JWF)、inverse discrete cosine transform(IDCT)、16-point Fast Fourier Transform(16-FFT)を用いた。クロック周期の値を1とし、各演算器の遅延の値は、各演算器の標準の遅延の値(加減算器:最大遅延0.95,最小遅延0.5 乗算器:最大遅延1.95,最小遅延0.95,シフト演算器最大遅延0.9,最小遅延0.45)を中心値とし、分散0.1の正規分布で与えた。各ベンチマークに対し、50組の遅延値設定を用意し、ストール数最小化実験を行った。ベンチマークに対するスケジューリングはリストスケジューリング、レジスタ割り当てはレフトエッジアルゴリズムにて行っている。表6.2, 6.3, 6.4に各ベンチマークに対する実験結果を示す。提案手法、クロック周期のみ調整、ストールのみ調整の3つの手法について、回路の総実行時間(クロック周期×ステップ数)の平均値、提案手法を基準とした同じ遅延値の設定での総実行時間の差の最大値,最小値,ストール数の平均の値(提案手法,ストールのみの調整),提案手法を基準とした同じ遅延値の設定でのストール数の差の最大値,最小値について評価を行った。

提案手法とストールのみを調整する手法とを比較したとき、すべてのベンチマーク対して、提案手法の平均総実行時間の値が小さく、またストール数も少なくなっている。これはスキューによる調整が有効に働いているためである。またストール数の差の最大値については、提案手法とストールのみの調整において、最小6(JWF)、最大13(IDCT, 16-FFT)の差が出たが、差の最小値については、与えられた遅延値が、ストールなしでセットアップ・ホールド条件を満たしているケースがあり、ストール数が共に0となって、差の最小値も0となっている。

一方、提案手法とクロック周期のみを調整する手法を比較した場合、すべてのベンチマークにおいて平均実行時間の値が提案手法において悪くなっている。今回の実験では最大遅延のばらつきによる伸びが比較的小さく、わずかなクロック周期の増加でタイミング条件が満たされること多かったために、クロック周期 T_c 単位で総実行時間が延びるストールが劣る結果となった。また制御タイミングスキューはレジスタ毎に設定されるため、入力と同じレジスタに出力を書き戻すような演算に対しては、スキューは機能せず、必ずストールが発生してしまうという問題もある。こうしたことからデータのレジスタへの割り当て方によって、必要となるストール数が変化することが容易に想像でき、提案手

表 6.1: ベンチマーク回路

ベンチマーク回路	ステップ数	演算数	演算器数	レジスタ数
JWF	11	17	3(alu:2,mul:1)	7
ICDT	20	67	7(alu:3,sht:2,mul:1)	24
16-FFT	21	81	6(alu:,4mul:2)	17

表 6.2: JWF

手法	平均総実行時間	総実行時間差の最大値	総実行時間差の最小値	平均ストール数	ストール数差の最大値	ストール数差の最小値
提案手法	11.86	0	0	0.86	0	0
クロック周期調整	11.26	0	-4.12	-	-	-
ストールのみ調整	13.22	6	0	2.22	6	0

法が有効に機能するようなデータのレジスタへの割り当て手法は今後の重要な課題となっている。

今回の実験では、演算器ごとに遅延の値を割り当て、配線遅延を考慮していないが、本来はレジスタ間の配線遅延も考慮する必要がある、今後、配線遅延を考慮した実験を行う必要がある。

表 6.3: IDCT

手法	平均総実行時間	総実行時間差の最大値	総実行時間差の最小値	平均ストール数	ストール数差の最大値	ストール数差の最小値
提案手法	24.5	0	0	4.5	0	0
クロック周期調整	21.17	0	-9.2	-	-	-
ストールのみ調整	27.38	13	0	7.38	13	0

表 6.4: 16-FFT

手法	平均総実行時間	総実行時間差の最大値	総実行時間差の最小値	平均ストール数	ストール数差の最大値	ストール数差の最小値
提案手法	26.18	0	0	5.18	0	0
クロック周期調整	22.29	0	-12.16	-	-	-
ストールのみ調整	32.1	13	0	11.1	13	0

第7章 まとめと今後の課題

本稿では、LSI 製造後に制御タイミングスキューとストールを調整する LSI の回路方式を提案し、その回路方式においてストール数を最小化する手法を提案した。これにより、信号伝播遅延の製造時ばらつきに対して、クロック周期の値を変更することなくかつ実行時間の伸びを低く抑えて、回路を正しく動作させることが可能となる。

本研究で行った評価実験では、スキュー調整機構とストール調整機構を導入することで、ストール調整のみと比較して、全般的に総実行時間の値を低くすることができた。しかしながら、クロック周期の調整を行う場合と比較した場合、提案回路手法の方が総実行時間の値が大きくなっている。これは、データパス回路において、入力と同じレジスタに演算結果を書き戻すとき、同レジスタ間の演算になるのでスキューによる調整ができず、ストールが行われたことと、今回の評価実験で正規分布に従って乱数で与えた遅延量が、与えられたクロック周期より大きな値になったとしても、差分の値は微小であり、クロック周期の調整量が微小であったためである。遅延ばらつきが、ある一箇所の演算のみ著しく値が増大した場合、提案回路手法の方が総実行時間の値が低くなる可能性がある。

このようなことから、本稿にて提案した回路方式が最も効率的に機能するようなデータパス合成は今後の重要な課題である。具体的には、スキュー調整機構が活かされるように入力と同じレジスタへの演算結果の書き戻しを少なくするようなレジスタ割り当てをする合成を検討する必要がある。また、このようなレジスタ割り当てを考えた場合、レジスタ数をどれだけ必要とするかも検討する必要がある。レジスタ数を無制限に使用できた場合、レジスタの書き戻しをなくすことができ、ストール数を減少させることができるが、回路規模が増大し、生産コストも高くなってしまう。このことから、提案回路方式が効率的にかつ、なるべくレジスタ数を抑えるレジスタ割り当てについても検討する必要がある。

また、提案回路手法であるスキュー調整機構とストール調整機構を実現する具体的回路についても検討する必要がある。スキュー調整機構については製造後にスキュー量が調整可能な素子 PDE(programmable delay element) が研究されている [7]。スキュー調節量に関しては実際に調整できる分解能や調整範囲があると考えられるので、そのような制約でも提案回路方式を用いて回路が正しく動作するよう調整する手法も考慮する必要がある。また同様にストール調整機構についても具体的な回路を検討する必要がある。

また、本研究で提案したスキュー・ストール調整手法は、製造後の回路の遅延量が得られたことが前提となっている。このため、製造後の回路の遅延量を測定する必要があり、その測定方法、または、製造後の遅延量が得られなくても、回路が正しく動作するようス

キュー・ストールを調整する手法を検討する必要がある。

最終的には、本研究で提案した回路手法を用いて、提案回路手法を活かした高位合成を行うことが考えられる。

謝辞

本研究を進めるにあたり，北陸先端科学技術大学院大学 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同助教 岩垣剛氏，博士課程小畑貴之氏，井上恵介氏，研究室の皆様にも感謝いたします．

参考文献

- [1] Paul S. Zuchowski, Peter A. Habitz, Jerry D. Hayes, Jeffery H. Oppold, "Process and Environmental Variation Impacts on ASIC Timing", Proc. International Conference on Computer Design(ICC'D), pp.336-342, 2004.
- [2] Jongyoon Jung and Taewhan Kin, "Timing Variation-Aware High-Level Synthesis", In Proc.ICCAD, pp.424-428, 2007.
- [3] 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遅延解析手法", 信学技報, pp.7-12, 2001.
- [4] K.Inoue, M.Kaneko, and T.Iwagaki, "Safe clocking register assignment in datapath synthesis", Proc. International Conference on Computer Design(ICC'D), pp.120-127, October 2008.
- [5] Jose Luis Neves and Edy G. Friedman, "Optimal Clock Skew Scheduling Tolerant to Process Variations", Conference on Design Automation(DAC), pp.623-628, 1996.
- [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クュー手法", 信学技報, pp.7-12, 2008
- [7] 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クュー手法", 信学技報, pp.13-18, 2007
- [8] 藤原 秀雄, デジタルシステムの設計とテスト, 工学図書株式会社, 2004.