

データパス合成における最小遅延補正問題の計算複雑度とアルゴリズム

井上 恵介[†] 金子 峰雄[†] 岩垣 剛[†]

[†] 北陸先端科学技術大学院大学 情報科学研究科 〒 923-1292 石川県能美市旭台 1-1

E-mail: †{k-inoue,mkaneko,iwagaki}@jaist.ac.jp

あらまし 半導体プロセスの微細化に伴い、寄生素子の影響増大、電圧の微弱化が進み、雑音や遅延ばらつきの問題が顕在化してきている。遅延ばらつきの下で回路がホールド条件を満足するための手法として、演算器の非クリティカルパスに遅延素子を挿入することでレジスタ間の最小遅延を補正する手法が考えられる。本稿ではデータパス合成における、最小遅延を補正する演算器数の最小化問題について検討し、全体の演算器数を変数とする場合この問題がNP困難のクラスに属することを示す。また、全体の演算器数を定数とする場合について多項式時間アルゴリズムを導いている。

キーワード データパス合成, レジスタ割り当て, 遅延ばらつき, 最小遅延補正

Complexities and Algorithms of Minimum-Delay Compensation Problems in Datapath Synthesis

Keisuke INOUE[†], Mineo KANEKO[†], and Tsuyoshi IWAGAKI[†]

[†] School of Information Science, Japan Advanced Institute of Science and Technology

1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan

E-mail: †{k-inoue,mkaneko,iwagaki}@jaist.ac.jp

Abstract As the feature size of VLSI becomes smaller, delay variations become a serious problem in VLSI design. The “setup” timing constraint can be fulfilled by choosing a clock period appropriately, while the “hold” timing constraint can not, and in many cases, the hold constraint becomes critical for a correct latch of a signal under delay variations. An approach to ensure the hold constraint under delay variations is to enlarge the minimum path delay between registers. It can be done by inserting delay elements on non-critical paths mainly in a functional unit. We call it “minimum path delay compensation” in this paper. This paper discusses the RT-level optimization problem to minimize the number of functional units which require minimum path delay compensation in datapath synthesis, and computational complexities and computation algorithms for this problem are revealed and proposed.

Key words Datapath synthesis, register assignment, delay variation, minimum-delay compensation

1. ま え が き

半導体プロセスの微細化に伴い、製造時に生じる物理的パラメータの変動(トランジスタのサイズ, 膜厚, 不純物密度のばらつき, 配線形状のばらつき等)や動作時の変動(電源電圧変動, 温度変化, クロストーク等)に起因する遅延ばらつきの問題が深刻化している。

遅延ばらつきの問題に対して異なる立場から様々な研究が行われている。代表的なものとして、設計マージンを削減するために遅延ばらつきを正確に見積もる研究[1], PDE (programmable delay element) を使用し製造後のLSIの遅延を調整する研究[2]等が挙げられる。遅延ばらつきの下で信号をラッ

チする際、セットアップ条件だけではなくホールド条件がクリティカルになることも多い。このため、ホールド条件を満たすために特別な構成をしたレジスタが提案されている[3]。また、遅延ばらつきに対する耐性を有するデータパスの新しいクラスとして構造的遅延変動耐性が提案されている[4]。構造的遅延変動耐性を有するデータパスは、有効計算時間を犠牲にすることなくホールド条件のための大きな設計マージンを確保できる。一方、遅延ばらつきの下でホールド条件を満たすためのその他の手法として、レジスタ間の最小パス遅延を増加させる手法が考えられる。これは主に演算器の非クリティカルパス上に遅延素子を挿入することで実現される。本稿ではこの手法を最小遅延補正と呼ぶ。

本稿では、データパスにおいて最小遅延補正を要する演算器数の最小化問題について論じる。結果として合成されるデータパスは、(1) 最小遅延補正と構造的遅延変動耐性に基づくレジスタ割り当ての少なくとも一方によって各演算の遅延ばらつきに対する耐性が保証され、(2) 最小遅延補正を要する演算器数とデータ保持に必要なレジスタ数が最適化されたデータパスである。本稿では、始めに全体の演算器数を変数とする場合この問題がNP困難のクラスに属することを明らかにし、次いで全体の演算器数を定数とする場合の多項式時間アルゴリズムを提案する。

本稿は次のように構成される。第2章で遅延ばらつきを考慮した設計の問題を、また第3章でホールド条件のための機能的特質を述べる。第4章と第5章にて、最小遅延補正と構造的遅延変動耐性に基づいたレジスタ割り当てとその適用例について述べる。第6章で結論と今後の課題について述べる。

2. 遅延ばらつきを考慮した設計の問題

本稿ではアルゴリズムレベル動作記述としてのデータフローグラフを入力とし、レジスタ転送レベルの構造記述と動作記述を出力する高位合成問題を扱う。データフローグラフの頂点は演算を表し、有向枝は演算間のデータ依存関係を表す。以降の議論において、 $O_a, O_{a'}, O_b, O_{b'}, \dots$ は演算を表し、 a, a', b, b', \dots はそれぞれ $O_a, O_{a'}, O_b, O_{b'}, \dots$ の出力データを表すものとする。

2.1 セットアップ条件とホールド条件

図1は、 O_a の出力データ a をレジスタ Reg 1 に書き込み、 a を用いて演算器 FU_A に割り当てられた O_b を実行し、 O_b の出力データ b をレジスタ Reg 2 に書き込む場合のタイミングを説明するものである。本稿ではデータパスは名目上ゼロスキューの下で設計されていると仮定する。すなわち、クロック源から Reg i における j 番目のフリップフロップ (FF) への遅延を $r(i, j)$ としたとき、全ての i, j に対して $r(i, j) = r_0$ とする (r_0 は定数)。また、クロック周期を t_c とする。Reg 1 から Reg 2 への最大パス遅延に Reg 1 のデータ出力遅延時間と Reg 2 のセットアップ時間を加えたものを d_{\max} とする。Reg 1 から Reg 2 への最小パス遅延に Reg 1 のデータ出力遅延時間を加え Reg 2 のホールド時間を引いたものを d_{\min} とする。 O_i の出力データをレジスタに書き込むクロックに同期した離散的なタイミングを $\sigma(i) \in \mathbb{Z}$ とする。

b をラッチするための Reg 2 への制御信号の到着時刻は b の Reg 2 への到着時刻よりも遅くなければならない。これをセットアップ条件という。

$$\sigma(a) \cdot t_c + d_{\max} \leq \sigma(b) \cdot t_c \cdots \text{セットアップ条件}$$

なお、表記の簡単化のために、 r_0 の値に関わらず、制御信号がレジスタに到着する名目上の時刻が t_c の整数倍となるように時刻原点を設定している。

一般にレジスタは複数のデータによって共有される。 a と a' が Reg 1 を共有し a のライフタイムが a' のライフタイムに先行すると仮定する。このとき、データ b をラッチするための

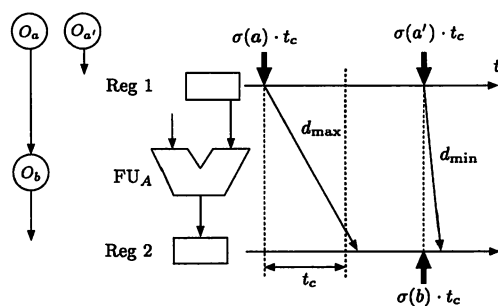


図1 セットアップ条件とホールド条件

Reg 2 への制御信号の到着時刻は b が破壊される時刻よりも早くなければならない。これをホールド条件という。

$$\sigma(b) \cdot t_c < \sigma(a') \cdot t_c + d_{\min} \cdots \text{ホールド条件}$$

本稿では、クロック信号 (または制御信号) のレジスタへの到着時刻のばらつきとデータパス上のパス遅延のばらつきを共に考慮する。 d_{\max} と d_{\min} の遅延変動量をそれぞれ $\Delta_{d_{\max}}$ と $\Delta_{d_{\min}}$ とする。 $r(i, j)$ の遅延変動量を $\Delta_{r(i, j)}$ とし、 $\Delta_{r(i) \min} = \min_j \{\Delta_{r(i, j)}\}$, $\Delta_{r(i) \max} = \max_j \{\Delta_{r(i, j)}\}$ とする。このとき、遅延ばらつきの下でのセットアップ条件とホールド条件は以下のように記述できる。

$$\begin{aligned} \sigma(a) \cdot t_c + \Delta_{r(1) \max} + d_{\max} + \Delta_{d_{\max}} &\leq \sigma(b) \cdot t_c + \Delta_{r(2) \min} \\ \sigma(b) \cdot t_c + \Delta_{r(2) \max} &< \sigma(a') \cdot t_c + \Delta_{r(1) \min} + d_{\min} + \Delta_{d_{\min}} \end{aligned}$$

但し、ここではタイミングをワード単位で一括して捉えており、これらは厳密にはセットアップ条件、ホールド条件を満たすための十分条件となっている。

2.2 機能的特質と性能的特質

個々の演算についてセットアップ条件やホールド条件が満たされるか否かは、採用するテクノロジーや下位の設計 (論理設計やレイアウト設計等) に強く依存する。また、高位の設計段階においては遅延および遅延ばらつきの正確な見積もりは困難であると考えられる。更に、IP (intellectual property) に基づく設計においては、遅延および遅延ばらつきは容易に、かつ大幅に変化し得る。本研究では、高性能データパスを設計するために、最初に機能的特質に優れたデータパスを合成し次に論理合成やレイアウトにおいてセットアップ条件とホールド条件を満たすための性能の最適化を行う設計方式を想定し、本稿ではこのうちのデータパスの機能的特質について議論する。

最大パス遅延のばらつきに起因するセットアップ条件違反はクロック周期の伸長や設計マージンの拡大によって解決できる。一方、最小パス遅延のばらつきに起因するホールド条件違反は一般的にそのような手法による解決ができない。本稿では遅延ばらつきを考慮したホールド条件を満たすためのデータパスの機能的特質について議論する。

3. ホールド条件のための機能的特質

3.1 最小遅延補正

遅延ばらつきを考慮したホールド条件により、

$$(\sigma(b) - \sigma(a')) \cdot t_c < d_{\min} + \Delta_{d_{\min}} + \Delta_{r(1)\min} - \Delta_{r(2)\max} \quad (1)$$

が要求される。遅延ばらつきの下でホールド条件を満たす手法の一つとして、 d_{\min} を遅延変動量の最悪値より大きくする手法が考えられる。これは主に演算器の非クリティカルパスに遅延素子を挿入することで実現される。本稿ではこの手法を最小遅延補正と呼ぶ。

3.2 構造的遅延変動耐性に基づいたレジスタ割り当て

遅延ばらつきに耐性を有するデータパスの新しいクラスとして構造的遅延変動耐性が提案された [4]。

構造的遅延変動耐性： 任意の有限な遅延ばらつきの下で、全ての演算（あるいはある特定の演算）に対して遅延ばらつきを考慮したセットアップ条件とホールド条件を満たすクロック周期が存在する（また、そのような回路）。 □

遅延ばらつきを考慮したホールド条件を満たすクロック周期が存在するための条件は次のように書くことができる。

C2'-1: O_b の入力データが b と異なるデータ a' によって上書きされるとき、 $\sigma(b) - \sigma(a') \leq -1$ 。

C2'-2: O_b の入力データが存在したレジスタに b を書き戻すことは許される。 $\sigma(b) - \sigma(a') = \sigma(b) - \sigma(b) = 0$ 。

なお、C2'-2 を正当ならしめる厳密な条件は $d_{\min} + \Delta_{d_{\min}} + \Delta_{r(1)\min} - \Delta_{r(1)\max} > 0$ と書ける。これは一つのレジスタを構成する FF 群に対して、最も遅く到着する制御信号と最も早く到着する制御信号との時間差が $d_{\min} + \Delta_{d_{\min}}$ 未満であることを意味している。

条件 C2'-1 と C2'-2 はデータのレジスタ割り当てと直接的な関係を有している。データ a のライフタイムの開始時刻は $(\sigma(a) \cdot t_c)_+$ であり、終了時刻は

$$\left(\max_{\text{for operation } O_b \text{ which uses data } a \text{ as input}} \{ \sigma(b) \cdot t_c \} \right)_-$$

である。但し、 t_+ は時刻 t の“直後”、 t_- は t の“直前”を表している。従来のレジスタ割り当てにおいてはライフタイムが重ならないデータ同士はレジスタ共有可能であると考え、一方、C2'-1 と C2'-2 を満たすためのレジスタ共有の可否はデータ間の依存関係に基づいて二つのタイプに分類される。

C2'-1: a を入力データとして最後に使用する演算 O_b の出力データ b と a' が異なる場合、開始時刻が $(\sigma(a') \cdot t_c)_+$ である a' と終了時刻が $(\sigma(a) \cdot t_c)_-$ である a はレジスタ共有できない。終了時刻が $((\sigma(a') - 1) \cdot t_c)_-$ 以前のデータと a' はレジスタ共有できる (図 2)。

C2'-2: a を入力データとして最後に使用する O_b の出力データ b と a はレジスタ共有できる (図 3(a))。しかし、 a を最後に使用する演算が 2 つ以上存在する場合、それらの出力データのいずれも a とレジスタ共有できない (図 3(b))。もしも、 a を入力データとして最後に使用する演算を O_b 、 $O_{b'}$ とし $(\sigma(b') = \sigma(b))$ 、 O_b に関する C2'-2 によって b と a がレジスタを共有すると、演算 $O_{b'}$ に条件 C2'-1 を適用する必要がある

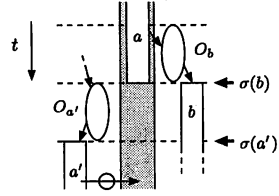


図 2 C2'-1 によるデータのレジスタ共有可能性。矩形はデータのライフタイム、楕円はスケジュールされた演算、灰色の領域はレジスタを表す。同じ灰色の領域に描かれたデータは同じレジスタに書き込まれるデータの集合を表す。

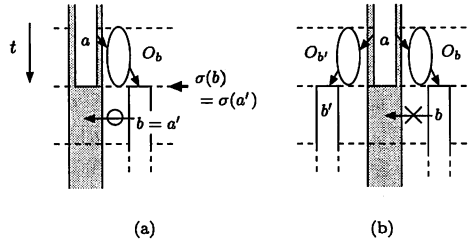


図 3 C2'-2 によるデータのレジスタ共有可能性。(a) O_b が a を最後に使用する唯一の演算であるとき、 a と b はレジスタ共有可能。(b) a を最後に使用する演算が他にも存在する場合、それらの出力データと a はレジスタ共有不可能。

(C2'-1 において O_b を $O_{b'}$ に、 b を b' に、 a' を b にそれぞれ置き換える)。この結果、 $\sigma(b') - \sigma(b) \leq -1$ が成り立たなければならないがこれは仮定に矛盾する。

4. 最小遅延補正と構造的遅延変動耐性に基づいたレジスタ割り当て

最小遅延補正では遅延素子の挿入に起因する面積の増加や電力消費量の増加、最大パス遅延の増大等の問題が生じる。一方、構造的遅延変動耐性に基づくレジスタ割り当ては、従来のレジスタ割り当てと比較してレジスタ数が増加する傾向がある。

4.1 問題の定式化と計算複雑度

ここで考える最小遅延補正と構造的遅延変動耐性に基づくレジスタ割り当て問題は以下のように定式化される。演算の集合を \mathcal{O} 、演算間の依存関係の集合を \mathcal{A} とするデータフローグラフ $G = (\mathcal{O}, \mathcal{A})$ 、データの集合 \mathcal{D} (\mathcal{O} と一対一対応を持つ)、制御信号スケジュール σ 、演算器集合 \mathcal{F} 、レジスタ集合 \mathcal{R} 、演算器割り当て $\rho: \mathcal{O} \rightarrow \mathcal{F}$ を入力とし、レジスタ割り当て $\xi: \mathcal{D} \rightarrow \mathcal{R}$ と最小遅延補正を施す演算器集合 $C \subseteq \mathcal{F}$ を出力とする。なお個々の演算は最小遅延補正または構造的遅延変動耐性に基づくレジスタ割り当てにてホールド条件が保証されるものとする。また、最適化目標を $|C|$ (最小遅延補正を要する演算器数) の最小化とする。

以下に示す二つの定理が本稿の主要な結果である。

定理 1: 演算器数に変数の場合、最小遅延補正と構造的遅延変動耐性に基づくレジスタ割り当て問題は NP 困難である。

定理 2: 演算器数が定数の場合, 最小遅延補正と構造的遅延変動耐性に基づくレジスタ割り当て問題はクラス \mathbf{P} に属する.

4.2 演算器数を変数とする場合の NP 困難性

定理 1 の証明は, NP 完全であることが知られている SET_PACKING [5] から判定問題版としての最小遅延補正と構造的遅延変動耐性に基づくレジスタ割り当て問題 ($|C| \leq K$ となる ξ , C が存在するか) への多項式時間還元を示すことで構成される. SET_PACKING は入力を集合族 S_1, S_2, \dots, S_n と正整数 K' とし, ある $I \subseteq \{1, 2, \dots, n\}$, $|I| \geq K'$ が存在して $\forall i \neq j \in I: S_i \cap S_j = \phi$ が成り立つか否かを問う問題である.

$\bigcup_{i=1}^n S_i = \{l_1, l_2, \dots, l_m\}$ とする. それぞれの i ($1 \leq i \leq m$) に対し l_i が S_1, \dots, S_n の中で現れる回数を g_i とする. また, $g_{\max} = \max_{1 \leq i \leq m} g_i$ とする.

O の定義 $O = \{O_{(i,j)} \mid 1 \leq i \leq m, 1 \leq j \leq g_i\} \cup \{O'_{(i,j)} \mid 2 \leq i \leq m, g_{i-1} \neq 1, 1 \leq j \leq g_{i-1} - 1\} \cup \{O_{(i,j)}^{(i+1)}, O'_{(i+1,j)} \mid 1 \leq i < m, 1 \leq j \leq g_{\max} - g_i\} \cup \{O_{(m,j)}^{(m+1)} \mid g_{\max} \neq g_m, 1 \leq j \leq g_{\max} - g_m + 1\}$ とする. 全ての演算は同一種類の演算器で実行され, 実行ステップ数 1 とする. $O_{(i,j)}$, $O'_{(i,j)}$, $O_{(i,j)}^{(k)}$ の出力データを $d_{(i,j)}$, $d'_{(i,j)}$, $d_{(i,j)}^{(k)}$ とする.

A の定義 $A = \{(O_{(i,1)}, O_{(i+1,j)}) \mid 1 \leq i < m, 1 \leq j \leq g_{i+1}\} \cup \{(O_{(i,j+1)}, O'_{(i+1,j)}) \mid 1 \leq i < m, g_i \neq 1, 1 \leq j \leq g_i - 1\} \cup \{(O'_{(i,j)}, O'_{(i+1,j)}) \mid 2 \leq i < m, 1 \leq j \leq g_{\max} - 1, j \leq g_i - 1, j \leq g_{i-1} - 1\} \cup \{(O_{(i,j)}^{(i+1)}, O'_{(i+1,j)} \mid 1 \leq i < m, g_{\max} \neq g_i, 1 \leq j \leq g_{\max} - g_i\}$ とする. 外部入力データを $e_1, e'_1, e_j^{(k)}$ とする. 全ての j ($1 \leq j \leq g_1$) に対して, $O_{(1,j)}$ は e_1 を入力データとする. 全ての j ($1 \leq j \leq g_{\max} - 1$) に対して i を $j \leq g_{i-1} - 1$ を満たす最小の i としたとき, $O'_{(i,j)}$ は e'_j を入力データとする. 全ての i ($1 \leq i < m$), j ($1 \leq j \leq g_{\max} - g_i$) に対して $O_{(i,j)}^{(i+1)}$, $O'_{(i+1,j)}$ は $e_j^{(i+1)}$ を入力データとする. 全ての j ($1 \leq j \leq g_{\max} - g_m + 1$) に対して $O_{(m,j)}^{(m+1)}$ は $e_1^{(m+1)}$ を入力データとする. 外部入力データのライフタイムは計算開始制御ステップ (ステップ 0) から最後に使用される制御ステップまでとする. ある演算の出力データ d が他の演算の入力データとして使用されない場合, d は計算終了までレジスタに保持されるデータとする.

σ の定義 $\sigma(O_{(i,j)}) = \sigma(O'_{(i,j)}) = \sigma(O_{(i,j)}^{(k)}) = i$.

F の定義 $F = \{F_j \mid 1 \leq j \leq n\} \cup \{F'_j \mid 1 \leq j \leq g_{\max} - 1\} \cup \{F_j^{(i)} \mid 2 \leq i \leq m + 1, 1 \leq j \leq g_{\max} - g_{i-1}\}$.

R の定義 h を $g_{\max} = g_m$ のとき -1 , そうでないとき 0 とする. $R = \{\text{Reg } i \mid 1 \leq i \leq (m+1)g_{\max} - \sum_{i=1}^{m-1} g_i + h\}$.

ρ の定義 それぞれの i ($1 \leq i \leq m$) に対し l_i が S_1, \dots, S_j ($j \leq n$) の中で現れる回数を $g_{(i,j)}$ とする. $1 \leq j \leq n, l_i \in S_j: \rho(O_{(i,g_{(i,j)})}) = F_j$, $2 \leq i \leq m, g_{i-1} \neq 1, 1 \leq j \leq g_{i-1} - 1: \rho(O'_{(i,j)}) = F'_j$, $1 \leq i < m, 1 \leq j \leq g_{\max} - g_i: \rho(O_{(i,j)}^{(i+1)}) = \rho(O'_{(i+1,j)}) = F_j^{(i+1)}$.

K の定義 $K = n - K'$ とする.

図 4 に変換の具体例を示す. 変換後の本問題に対する入力インスタンスに関して以下の補題群が成り立つ.

補題 1: それぞれの i ($i = 1, 2, \dots$) に対して, $d_{(i,1)}$ と, $d_{(i,1)}$

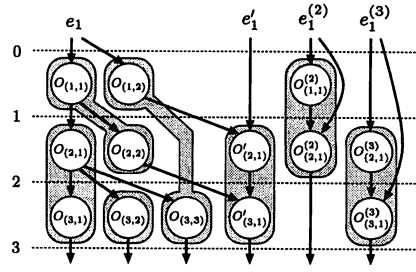


図 4 SET_PACKING の入力インスタンス $\{1, 2\}$, $\{2, 3\}$, $\{3\}$, $\{1, 3\}$ を本問題への入力インスタンスに変換した例 ($l_i = i$ とする). 点線は制御ステップ, 円は演算, 有向枝は演算間のデータ依存関係を表す. 灰色の領域で囲まれた演算は同じ演算器に割り当てられていることを表す.

を最後に使用する演算の出力データを同一のレジスタに割り当てると規定しても最適性を損わない.

補題 2: 任意の ξ に対して, 最小遅延補正を必要としない演算器は, 一入力演算のみが割り当てられた演算器である.

補題 3: 任意の ξ と制御ステップ i に対して, i にスケジュールされる一入力かつ非補正演算は高々一つしか存在しない.

定理 1 の証明: 本問題が NP 問題であることと変換手続きの計算複雑度が入力演算器数の多項式オーダーで可能であることは明らかである.

SET_PACKING の yes インスタンスを考える. $\rho(O_{(1,j)}) = F_j$, $j \in I$ ならば e_1 と $d_{(1,j)}$ を同一のレジスタに割り当てると規定し, マージして一つのライフタイムとする. 全ての i ($i = 2, 3, \dots$) に対して $\rho(O_{(i,j)}) = F_j$, $j \in I$ ならば, $d_{(i-1,1)}$ と $d_{(i,j)}$ を同一のレジスタに割り当てると規定し, マージして一つのライフタイムとする. 補題 1 により, この決定は最適性を損わない. 新しく得られたライフタイムの集合に対してレフトエッジアルゴリズム [6] を適用した結果のレジスタ割り当てを ξ とする. $\forall i \in I: F_i \in \mathcal{F} - C$ であることから, $|C| \leq n - |I| \leq K$ が成り立つ.

次に, 変換後のインスタンスが本問題の yes インスタンスであるとする. $I = \{i \mid F_i \in \mathcal{F} - C\}$ とする. $\exists i \neq j \in I: l_k \in S_i \cap S_j$ と仮定すると, 補題 2 により制御ステップ l_k にスケジュールされる一入力かつ非補正演算が二つ以上存在するが, これは補題 3 に矛盾する. 従って, $\forall i \neq j \in I: S_i \cap S_j = \phi$ が成り立つ. また, $|I| = n - |C| \geq K'$ が成り立つ. ■

4.3 演算器数を定数とする場合の多項式時間アルゴリズム

最初に提案アルゴリズムの核となるサブルーチンについて述べる. このサブルーチンは C を入力とし, ξ を出力する.

与えられた最小遅延補正割り当てに対する構造的遅延変動耐性に基づくレジスタ割り当て (Sub_Reg_Assign):

手順 1 データ d を最後に使用する演算の中で $\mathcal{F} - C$ に含まれている演算器で実行される演算 O が唯一存在するとき, d と O の出力データを同じレジスタを使用するものと規定しライフタイムをマージして一つのライフタイム (一つの区間) とする. この手続きを可能な限り繰り返し実行する.

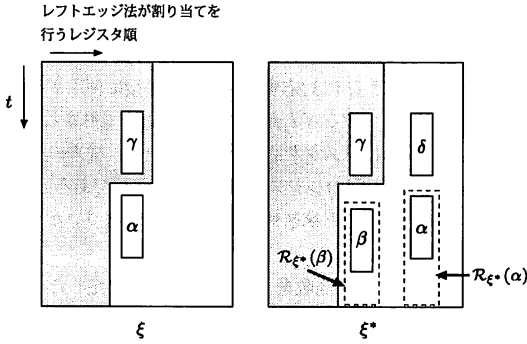


図5 レジスタ割り当て ξ と ξ^* . $\alpha, \beta, \delta, \gamma$ はデータを表す。灰色の領域は ξ と ξ^* に共通して割り当てられるデータの集合を表す。

手順 2. D を手順 1 にて得られた区間と、ライフタイムがマージされずに残ったデータの集合とする。 D の要素 d に対して d を最後に使用する演算の中に $\mathcal{F}-C$ に含まれる演算器で実行される演算が含まれる場合、 d のライフタイムの最終ステップに 1 ステップ加える。こうして得られたライフタイムの集合を拡張ライフタイムの集合と呼ぶ。

手順 3. 拡張ライフタイムの集合に対してレフトエッジアルゴリズムを適用する。 □

補題 4: Sub_RegAssign は与えられた最小遅延補正割り当てに対する構造的遅延変動耐性に基づくレジスタ割り当てにおいて、レジスタ数最小を保証する。

以下の議論において、データ d のライフタイムの開始時刻と終了時刻をそれぞれ $l_e(d)$, $l_s(d)$ とする。 $\mathcal{R}_\xi(d)$ を ξ において d と同じレジスタに割り当てられているデータ集合の中でライフタイムの開始時刻が $l_s(d)$ 以降であるデータ集合とする。

補題 4 の証明: ξ を Sub_RegAssign で得られたレジスタ割り当てとし、 ξ^* を与えられた最小遅延補正演算器におけるレジスタ数最小のレジスタ割り当てとする。 Sub_RegAssign の手順 3 におけるライフタイムの選択順と同じ順番で ξ と ξ^* に割り当てられたデータを比較していくことを考える。ライフタイムが、手順 1 において複数のライフタイムを合成したものであった場合、その中のライフタイムの順番は開始制御ステップが早いものを先とする。 $\xi \neq \xi^*$ と仮定すると、最初から比較したときに最後に共通するデータ γ が存在する。 ξ における γ の次のデータを α とし、 ξ^* における γ の次のデータを β とする ($\beta \neq \alpha$)。また、 δ を ξ^* における α の直前のデータとする (図 5)。レジスタ数最小となる解が複数存在する場合、最初から γ まで ξ と最も多く共通データを持つものを ξ^* とする。ここで、 ξ と ξ^* の状況についての場合分けを三つ組 (x, y, z) で表現する。 x, y, z はそれぞれ 0 または 1 をとる変数であり、その値は以下のように意味付けされる。

- x : ξ において $l_e(\gamma) = l_s(\alpha)$ のとき 0, そうでないとき 1.
 - y : ξ^* において $l_e(\gamma) = l_s(\beta)$ のとき 0, そうでないとき 1.
 - z : ξ^* において $l_e(\delta) = l_s(\alpha)$ のとき 0, そうでないとき 1.
- $(x, y, z) = (0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 1, 0), (1, 1, 1)$

の場合: ξ^* における $\mathcal{R}_{\xi^*}(\alpha)$ と $\mathcal{R}_{\xi^*}(\beta)$ を交換したレジスタ割り当てを ξ^{**} とする。 ξ^{**} は ξ^* とレジスタ数が等しく ξ^* よりも ξ と一つ多くのデータを共通に持つ。しかしこれは ξ^* の選択条件に反する。

$(x, y, z) = (1, 0, 0)$ かつ O_β が補正の場合、または $(x, y, z) = (1, 0, 1)$ かつ O_β が補正の場合: $l_s(\alpha) > l_s(\beta)$ であるにも関わらず ξ において γ の次に β が選択されないことは Sub_RegAssign に反する。

$(x, y, z) = (1, 0, 0)$ かつ O_β が非補正の場合、または $(x, y, z) = (1, 0, 1)$ かつ O_β が非補正の場合: $l_s(\alpha) > l_s(\beta)$ であるにも関わらず ξ において α よりも β が先に選択されないということは、あるデータ c が存在し $l_e(\gamma) = l_e(c)$ が成り立ち、 β が Sub_RegAssign によって c とマージされたことを意味する。 $c = \delta$ であれば、 ξ^* における $\mathcal{R}_{\xi^*}(\alpha)$ と $\mathcal{R}_{\xi^*}(\beta)$ を交換したレジスタ割り当てを ξ^{***} とする。 $c \neq \delta$ であれば、 ξ^* において c の直後に存在するデータを d とする。 ξ^* における $\mathcal{R}_{\xi^*}(\beta)$ と $\mathcal{R}_{\xi^*}(d)$ を交換したレジスタ割り当てを ξ^{**} とする。 ξ^{**} において $l_s(d) < l_s(\alpha)$ と仮定すると ξ において α ではなく d が先に選ばれるはずである。従って、 $l_s(d) \geq l_s(\alpha)$ である。 ξ^{**} における $\mathcal{R}_{\xi^{**}}(d)$ と $\mathcal{R}_{\xi^{**}}(\alpha)$ を交換したレジスタ割り当てを ξ^{***} とする。 ξ^{***} は ξ^* と比べて ξ と一つ多くのデータを共通に持つ。しかしこれは ξ^* の選択条件に反する。 ■

最小遅延補正の割り当て $\mathcal{C} \subseteq \mathcal{F}$ の全ての組み合わせに対して Sub_RegAssign を適用することにより、本問題の最適解を得るアルゴリズムを構成できる。

最小遅延補正と構造的遅延変動耐性に基づくレジスタ割り当て:

```

Let  $S = \phi$ : 最小遅延補正演算器集合とレジスタ割り当て;
for ( $\mathcal{F}$  の各部分集合  $C$ ) {
   $\xi \leftarrow \text{Sub\_RegAssign}(C)$ ;
  if ( $\xi$  のレジスタ数  $\leq$  入力レジスタ数)
    then  $S \leftarrow S \cup \{(\xi, C)\}$ ;
}
Output  $|C|$  が最小の  $(\xi, C) \in S$ ;

```

演算器数を定数とすれば提案アルゴリズムの計算複雑度は演算数の多項式オーダーであることと補題 4 から定理 2 が成り立つ。

5. 適用例

本節では、最小遅延補正と構造的遅延変動耐性に基づくレジスタ割り当ての適用例を示す。入力インスタンスとして、5 次楕円ウェーブデジタルフィルタベンチマーク [7] を用いる。データフローグラフを無閉路有向グラフとして扱うために遅延枝を切り開いて外部入力、外部出力とする。図 6, 図 7 において円は演算を表し、傍らの整数値 i はその演算が O_i であることを表す。アルファベットは中間データ、 $\text{inp } i$ は外部入力、 $\text{out } i$ は外部出力を表す。ハードウェア資源として加算器 3 個、乗算器 1 個を使用し、各演算の実行制御ステップ数を 1 とする。加算器の割り当てを $\rho(O_i) = \text{ADD}_1(i = 1, 2, 4, 5, 7, 9, 13, 16, 20, 23, 29, 30)$, $\rho(O_i) = \text{ADD}_2(i = 3, 10, 12, 14, 17, 21, 25, 26, 31, 33)$, $\rho(O_i) = \text{ADD}_3(i = 18, 22, 27, 34)$ とする。また、全ての乗

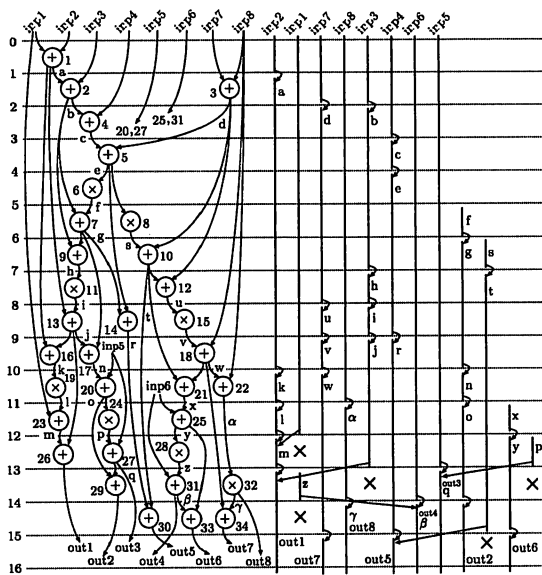


図6 構造的遅延変動耐性のためのスケジュールとレジスタ割り当て結果。記号“x”は、データを割り当てることができない制御ステップを表している。

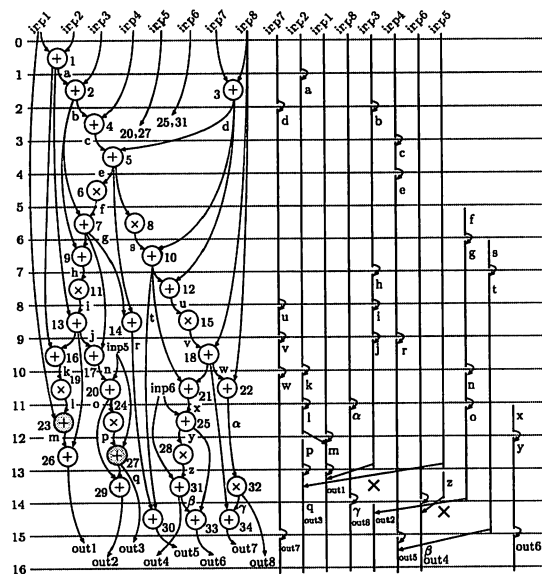


図7 使用可能なレジスタの総数が11の場合のスケジュールとレジスタ割り当て結果。最小遅延補正演算は O_{23} 、 O_{27} である。

算を一つの乗算器 MUL に割り当てる。

5.1 構造的遅延変動耐性に基づいたレジスタ割り当て

構造的遅延変動耐性に基づいたレジスタ割り当ての適用例を図6に示す。データバスに構造的遅延変動耐性を持たせるためのレジスタ数最小を保証する多項式時間アルゴリズムが知られている[4]。図6におけるタイムチャートの左図はスケジュールであり演算スケジュールとデータ間の依存関係を示しており、右図はレジスタ割り当て表である。右図における縦線分はデー

タのライフタイムを表し、同列に描かれたデータ同士は同じレジスタに割り当てられたことを表している。ライフタイム l_i から l_j への有向枝は l_i に対応するデータを最後に使用する演算の出力データのライフタイムが l_j であることを意味する。図6の例では12個のレジスタを必要とする。一方、従来のレジスタ割り当てでは11個のレジスタで割り当て可能である。

5.2 最小遅延補正と構造的遅延変動耐性に基づいたレジスタ割り当て

使用可能なレジスタ総数を入力として11と指定したときの最小遅延補正と構造的遅延変動耐性に基づいたレジスタ割り当てを図7に示す。スケジュールにおいて灰色の演算は、ホールド条件のためにその演算を実行する演算器が最小遅延補正を必要とする演算を表している。その他の演算に対するホールド条件は構造的遅延変動耐性に基づいたレジスタ割り当てによって保証される。図7の例では2個の加算器が最小遅延補正を必要とする。前節で示したように、レジスタを12個用意すれば、どの演算器にも最小遅延補正を施すことなく構造的遅延変動耐性に基づいたレジスタ割り当てを行うことができ、全ての演算に対するホールド条件を保証できることになる。

6. むすび

本稿では、最小遅延補正と構造的遅延変動耐性に基づくレジスタ割り当てによって遅延ばらつきに耐性を持つデータバスの合成を提案した。演算器数が変数の場合、最小遅延補正と構造的遅延変動耐性に基づくレジスタ割り当て問題はNP困難であり、演算器数が定数の場合クラスPに属することを示した。本稿では主にデータバスの機能的特質に注目して議論した。今後の課題として、機能的特質の下で性能的特質の最適化やヒューリスティックを用いた効率的な解法の開発が挙げられる。

謝辞 本研究は一部、日本学術振興会科学研究費補助金 基礎研究(C) (課題番号 19560340) の研究助成による。

文献

- [1] K. Nose, M. Kajita, and M. Mizuno, "A 1ps-resolution jitter-measurement macro using interpolated jitter oversampling," *Proc. International Solid-State Circuits Conference*, pp.2112-2121, February 2006.
- [2] E. Takahashi, Y. Kasai, M. Murakawa, and T. Higuchi, "A post-silicon clock timing adjustment using genetic algorithms," *Digest of technical papers of the 2003 symposium on VLSI circuits*, pp.13-16, February 2003.
- [3] J. Xi and W.W.-M. Dai, "Jitter-tolerant clock routing in two-phase synchronous systems," *Proc. the 16th International Conference on Computer-Aided Design*, pp. 316-320, 1996.
- [4] 井上 恵介, 金子 雄雄, 岩垣 剛, "データバスにおける遅延変動耐性に対する基礎的検討," 第20回回路とシステム軽井沢ワークショップ論文集, April 2007.
- [5] M. Garey and D. Johnson, *Computers and intractability - a guide to the theory of NP-completeness*, W.H. Freeman Publishers, 1979.
- [6] F.J. Kurdahi and A.C. Parker, "REAL: A program for register allocation," *Proc. the 24th Design Automation Conference*, pp.210-215, June 1987.
- [7] P. Michel, U. Lauther, and P. Duzy, *The synthesis approach to digital system design*, Kluwer Academic Publishers, 1992.