NAIST-IS-DT9761009

博士論文

順序回路の無閉路構造に基づく部分スキャン設計ならびに高位合成に関する研究

高崎 智也

2000年3月24日

奈良先端科学技術大学院大学
情報科学研究科 情報処理学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
博士 (工学) 授与の要件として提出した博士論文である。

高崎 智也

指導教官： 藤原 秀雄 教授
渡邊 勝正 教授
福田 晃 教授
増澤 利光 助教授

順序回路の無閉路構造に基づく部分スキャン設計ならびに高位合成に関する研究*

高崎 智也

内容梗概

近年のVLSIの高集積化，大規模化に伴い，回路に故障が存在するかどうかを調べるためのテストはますます重要で，かつ困難な問題となっている．テストの費用を削減するために，回路に余分なハードウェアを付加してテスト容易な回路を実現するテスト容易化設計法の研究開発が望まれている．順序回路の一部のフリップフロップをスキャン可能なフリップフロップ(スキャンフリップフロップ)に置き換える部分スキャン設計は，小さいハードウェアオーバヘッドでテスト容易な回路を実現する重要なテスト容易化設計法の一つである．一方，回路設計の初期の段階として，抽象度の高い動作記述からレジスタ転送レベル(RTL)の回路を合成する高位合成の段階でテスト容易性を考慮することにより，回路の面積・性能とともにテスト容易性も含めた最適化および設計費用の削減ができるものと期待されている．本論文では，部分スキャン設計にともなうハードウェア(面積)オーバヘッドが小さく，テスト容易な回路を実現する方式として，以下の2つの手法を提案する．

まずはじめに，順序回路の無閉路構造に基づく部分スキャン設計の一手法として，内部平衡構造に基づく拡張部分スキャン設計法を提案する．この拡張部分スキャン設計において，スキャン化にともなう面積オーバヘッドを小さくするための，フリップフロップと信号線を選択する方法を述べる．また，核回路を内部平衡構造とする拡張部分スキャン設計された回路について，核回路に対するテスト生成法とスキャン化により新たに付加した回路のテスト生成法について述べる．

*奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 博士論文，NAIST-IS-DT9761009，2000年3月24日．

ベンチマーク回路に対する実験結果では，提案した拡張部分スキャン設計が小さい面積オーバーヘッドで実現できることを示す．

次に，テスト容易性を考慮した高位合成 (テスト容易化高位合成) の一手法として，無閉路構造に基づく部分スキャン設計のためのデータパスのテスト容易化高位合成法を考察する．スケジュールされた動作記述 (データフローグラフ) に対して，面積 (リソース数) の最小性を満たしながら，無閉路化のためのスキャンレジスタ数を最小にする演算器とレジスタのバインディング法を提案する．本合成法は，テスト容易性を考慮しない従来手法と比較して，リソース数を増やすことなく，無閉路化のためのスキャンレジスタ数の小さいレジスタ転送レベルデータパスを合成することができる．

キーワード

順序回路，無閉路構造，部分スキャン設計，内部平衡構造，高位合成

Studies on Partial Scan Design and High-Level Synthesis Based on Acyclic Structure of Sequential Circuits*

Tomoya Takasaki

Abstract

The greater circuit density of VLSI makes testing more important and more difficult. In order to reduce the testing cost, design for testability (DFT) techniques, which implement an easily testable circuit by adding extra hardware for a circuit, have been expected. Partial scan design, which replaces a part of flip-flops in a sequential circuit by scannable flip-flops (scan flip-flops), is one of the important DFT techniques to implement an easily testable circuit with low hardware overhead. On the other hand, design cost reduction of VLSI's can be achieved by considering testability at the stage of high-level synthesis, which synthesizes a register-transfer-level (RTL) circuit from an abstract behavioral description, as the early stage of VLSI design. In this thesis, we propose the following two methods for implementing an easily testable circuit with low hardware (area) overhead by partial scan design.

First, we propose an extended partial scan design method based on internally balanced structure. We present an algorithm for finding a set of flip-flops and wires in order to minimize the area overhead for the extended partial scan design. We consider a test generation method for the circuits with the extended partial scan design. Experimental results for benchmark circuits show that the proposed extended partial scan design can be implemented with low area overhead.

*Doctor's Thesis, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT9761009, March 24, 2000.

Second, we present a high-level synthesis method for testable data paths with partial scan design based on acyclic structure. For a given scheduled behavioral description (data flow graph), we propose a method of operational unit and register bindings to minimize the number of scan registers for acyclic structure without sacrifice of area overhead (the number of resources). The synthesis method can generate register-transfer-level data paths with a small number of scan registers for acyclic structure without increasing the number of resources compared with previous methods without consideration to testability.

Keywords:

sequential circuits, acyclic structure, partial scan design, internally balanced strcture, high-level synthesis

関連文献一覧

- 学術論文誌

1. 藤原秀雄, 大竹哲史, 高崎智也, "組合せテスト生成複雑度でテスト生成可能な順序回路構造とその応用," 電子情報通信学会論文誌 DI, Vol.J80-D-I, No.2, pp.155-163, February 1997.
2. 高崎智也, 井上智生, 藤原秀雄, "内部平衡構造に基づく部分スキャン設計法の考察," 電子情報通信学会論文誌 DI, Vol.J81-D-I, No.3, pp.318-327, March 1998.
3. 高崎智也, 井上智生, 藤原秀雄, "無閉路部分スキャン設計に基づくデータパスのテスト容易化高位合成に おけるバインディング手法," 電子情報通信学会論文誌 DI, Vol.J83-D-I, No.2, pp.282-292, February 2000.

- 国際会議 (査読付き)

1. Tomoya Takasaki, Tomoo Inoue and Hideo Fujiwara, "Partial Scan Design Methods Based on Internally Balanced Structure," Proc. Asia and South Pacific Design Automation Conference 1998, pp.211-216, February 1998.
2. Tomoya Takasaki, Tomoo Inoue and Hideo Fujiwara, "A High-Level Synthesis Approach to Partial Scan Design Based on Acyclic Structure, " Proc. IEEE the 8th Asian Test Simposium, pp.309-314, November 1999.

- 研究会報告

1. 高崎智也, 藤原秀雄, "組合せテスト生成複雑度でテスト生成可能な順序回路," 電子情報通信学会総合大会, 情報・システム分冊 1, SD-2-3, pp.321-322, March 1996.
2. 高崎智也, 井上智生, 藤原秀雄, "組合せテスト生成可能な拡張部分スキャン設計," 信学技報, FTS96-41, Vol.96, No.291, pp.1-8, October 1996.

3. 高崎智也, 井上智生, 藤原秀雄, “無閉路部分スキャン設計を指向したデータパスのテスト容易化高位合成,” 信学技報, FTS98-114, Vol.98, No.488, pp.65-72, December 1998.

目 次

図目次

表目次

第 1 章

序 論

近年の半導体技術の急速な進歩に伴う VLSI の普及により，その信頼性は非常に重要な問題となっている．正しく動作しない VLSI の存在は，それを構成するシステムのサービスの停止，中断にとどまらず，社会的に大きな影響を与えることにもなりかねない．信頼性の高い VLSI を設計，製造するためには VLSI 回路に故障 (物理的欠陥) がないことを保証するテストが不可欠であるが，回路の大規模複雑化にともない，テストにかかる費用の増大が問題となっており，また，テストの質の向上が求められている．

論理 VLSI は論理回路としてモデル化でき，論理 VLSI 中の多くの故障は論理回路の論理機能が故障により別の論理機能に変化してしまう論理故障でモデル化できる．論理故障の中で最もよく使われる故障に，論理素子の入出力線 (信号線) の値が 0 または 1 に固定される縮退故障がある．本研究で対象とする故障モデルは，信号線の縮退故障とする．縮退故障に対するテストは，論理回路に入力パターン (入力系列) を印加し，その出力応答を観測し，期待値 (故障がない場合の正常な出力) と比較することで行われる．出力応答と期待値が異なるとき，その故障は検出されたという．故障を検出するための入力パターン (入力系列) を求めることをテスト生成といい，この入力パターン (入力系列) のことをテストパターン (テスト系列) と呼ぶ．また，テストパターン (テスト系列) を VLSI 回路に印加して，その出力応答を期待値と比較し，故障の有無を判定することをテスト実行と呼ぶ．

テストの費用は，テスト生成時間およびテスト実行時間で評価できる．また，

テストの質は生成されたテストパターン (テスト系列) 集合の**故障検出率**や**故障検出効率**で評価できる．故障検出率は，論理回路中のテスト生成の対象故障に対する，生成されたテストパターン (テスト系列) で検出できる故障の割合を示す．故障検出効率は，論理回路中のテスト生成の対象故障に対する，生成されたテストパターン (テスト系列) で検出できる故障とテスト生成アルゴリズムが冗長と判定した故障の割合を示す．

論理回路には，記憶素子 (フリップフロップ) を含まない組合せ回路とそれを含む順序回路がある．組合せ回路については，これまで効率のよいテスト生成アルゴリズムが提案されており [1, 2]，大規模な回路に対しても実用的なテスト生成時間で 100%の故障検出効率を得ることが可能である．一方，順序回路のテスト生成の複雑さはフリップフロップの内部状態数に依存し，回路規模が大きくなればテスト生成に膨大な時間がかかり，高い故障検出効率を得ることは困難となる．そのため，順序回路に対してはテスト生成を容易にし，かつ，故障検出効率を向上させるための設計変更 (**テスト容易化設計**) が必要である．

近年の設計自動化技術の進歩により，VLSI の機能・論理設計においては，レジスタ転送レベル (機能レベル) の記述からゲートレベルの論理回路を自動合成する**論理合成**の技術に関するさまざまな研究が行われており，この技術が実用化され，実際の設計に利用され，効果を上げ始めている [3]．また，機能設計の自動化のための手段として，アルゴリズムレベルの動作記述から性能やコストに関する制約を与えて，それを満たすレジスタ転送レベルの回路記述を自動的に生成する**高位合成**の技法が提案されている [3, 4]．一方，VLSI の設計においては，設計対象が，1) ゲートレベルの論理設計から論理合成を用いたレジスタ転送レベルでの機能設計への移行，2) 論理合成から高位合成への移行，というように，より抽象度の高い，上位のレベルからの設計自動化が進められている．そこで，テスト容易化設計においても，上位レベルでテスト容易化を考慮することにより，面積や動作速度とともにテスト容易性も含めた全般的な最適化および設計費用の削減が実現できるものと期待されている．

論理レベルまたはレジスタ転送レベルにおける順序回路に対する代表的なテスト容易化設計法として，**完全スキャン設計法** [1, 2] がある．完全スキャン設計では順序回路中のすべてのフリップフロップをスキャンフリップフロップ (テスト

の際に回路の外部から状態を任意に制御・観測できるフリップフロップ) に置き換える．これによりテスト生成の際にすべてのフリップフロップを外部入出力と考えることができるので，テスト生成の対象回路が順序回路中の組合せ回路部分となる．したがって，組合せ回路用のテスト生成アルゴリズムを適用することができ，短いテスト生成時間で100%の故障検出効率を得ることができる．しかし，完全スキャン設計では，すべてのフリップフロップをスキャンフリップフロップに置き換えるので，余分なロジックを付加することによって，面積オーバヘッドが大きくなるという問題がある．さらに，スキャンフリップフロップの値を設定・観測するのに，シフトレジスタとして動作させることで，テスト実行時間が長くなるという問題も生じる．

これらの問題を解決するため，順序回路中の一部のフリップフロップをスキャンフリップフロップに置き換える**部分スキャン設計法**が提案されている．部分スキャン設計では，テスト生成の対象回路は順序回路となる．文献[5, 6]では，順序回路用のテスト生成アルゴリズムを適用し，テスト生成の対象回路がセルフループ以外のフィードバックループを構成しないように，スキャンフリップフロップを選択する手法が提案されている．一方，部分スキャン設計で順序回路となるテスト生成の対象回路に対し，組合せ回路用のテスト生成アルゴリズムを適用する手法が提案されている [7, 8, 10]．これらの部分スキャン設計法において共通することは，順序回路中の一部のレジスタ (フリップフロップの組) をスキャンレジスタに置き換え，スキャンレジスタによってセルフループを含むすべてのフィードバックループを切断することにより，**無閉路構造**を実現する**無閉路部分スキャン設計**である．本研究で対象とする部分スキャン設計は，テスト生成の対象となる順序回路に組合せ回路用のテスト生成アルゴリズムを適用できる無閉路部分スキャン設計である．

本論文では，無閉路部分スキャン設計の一手法として，内部平衡構造に基づく拡張部分スキャン設計法を提案する．**内部平衡構造** [8] とは，無閉路構造の順序回路の一つで，**平衡構造** [7] を拡張した，組合せ回路用のテスト生成アルゴリズムでテスト生成可能な順序回路のクラスである．本部分スキャン設計法では，テスト生成の対象回路として，内部平衡構造の順序回路を考える．さらに，順序回路中の一部のフリップフロップだけでなく，信号線の値も任意に設定・観測でき

るように設計変更をする**拡張部分スキャン設計**の手法を取り入れ，テスト容易化にともなう面積オーバヘッドをより小さくすることを考える．

さらに本論文では，テスト容易性を考慮した高位合成 (**テスト容易化高位合成**) の一手法として，無閉路部分スキャン設計のスキャンレジスタ数 (面積オーバヘッド) を最小にするための高位合成法を提案する．高位合成の結果，生成されるレジスタ転送レベルの回路記述は，**コントローラ** (回路の制御部) と**データパス** (内部計算を行うデータ処理部) から構成されるが，ここではデータパスを対象としている．データパスの高位合成は，レジスタ転送レベルの回路で用いられるハードウェア要素 (演算器，レジスタなど) の型と個数を決定する**アロケーション**，動作記述中の演算をクロックに同期した制御ステップに割り当てる**スケジューリング**，動作記述中の演算を演算器に，変数をレジスタに割り当て，演算器とレジスタ間の相互接続を決定し，マルチプレクサやバスの割り当てを行う**バインディング**の 3 つのタスクから主に成り立っているが，ここでは高位合成の部分問題として，演算器とレジスタのバインディング法を提案する．本バインディング法を適用することにより，無閉路部分スキャン設計のスキャンレジスタ数は，高位合成の段階でテスト容易性を考慮しない従来手法と比較して，大幅な削減を期待することができる．

本論文の構成は以下の通りである．第 2 章では，順序回路の無閉路構造に基づく部分スキャン設計の一手法として，内部平衡構造に基づく拡張部分スキャン設計法を提案する．第 3 章では，テスト容易性を考慮した高位合成 (テスト容易化高位合成) の一手法として，無閉路構造に基づく部分スキャン設計のためのデータパスのテスト容易化高位合成法を提案する．第 4 章で研究のまとめと今後の課題について総括する．

第 2 章

内部平衡構造に基づく拡張部分スキャン設計法

2.1. はじめに

一般に組合せ回路に対しては，冗長故障を除いてほぼ100%の故障検出効率を達成するテスト生成アルゴリズムが存在する．一方，順序回路のテスト生成は困難な問題で，回路規模が大きくなれば解けなくなる場合が多い．この問題を解決するために，回路中のすべてのフリップフロップをスキャン可能なフリップフロップ (スキャンフリップフロップ) に置き換える完全スキャン設計法が提案されている [1, 2]．完全スキャン設計では，スキャンフリップフロップを取り除いた残りの回路 (核回路) が組合せ回路となるので組合せ回路のテスト生成アルゴリズムでテスト生成可能 (組合せテスト生成可能) となり，ほぼ100%の故障検出効率が得られるが，回路の面積オーバーヘッドが大きくなるという問題が生じる．順序回路中の一部のフリップフロップをスキャンフリップフロップに置き換える部分スキャン設計は小さい面積オーバーヘッドでテスト生成容易な回路を実現するための技術の一つである．しかし，順序回路となる核回路に対し，順序回路のテスト生成法を必要とする部分スキャン設計法 [5, 6] では，高い故障検出効率を達成するのが依然として困難である．これに対し，文献 [7] は組合せテスト生成可能な順序回路として平衡構造を提案し，核回路を平衡構造とする部分スキャン設計法

を示した．また，著者らは先に組合せテスト生成可能な順序回路として，平衡構造を拡張した**内部平衡構造順序回路**を提案し，その性質を利用して部分スキャン設計への応用について示した [8].

本章ではスキャン化による面積オーバヘッドをさらに小さくするために，順序回路内のフリップフロップに限らず，信号線をスキャンフリップフロップと同様の働きをするフリップフロップ [2, 11, 13](**バイパスフリップフロップ**) に置き換える**拡張部分スキャン設計**の手法を取り入れ，核回路を内部平衡構造とする拡張部分スキャン設計法を提案する．この拡張部分スキャン設計において，スキャン化による面積オーバーヘッドを小さくするフリップフロップと信号線を選択する方法を述べる．また，核回路を内部平衡構造とする拡張部分スキャン設計された回路について，核回路に対するテスト生成法とスキャン化により新たに付加した回路のテスト生成法について述べ，これらのテスト生成法を用いて生成される系列が正しいテスト系列であることの正当性を示す．さらに，ISCAS'89 ベンチマーク回路に対する実験結果より，提案した拡張部分スキャン設計が小さい面積オーバーヘッドで実現できることを示す．

2.2. 組合せテスト生成可能な順序回路

順序回路にフィードバックループがあれば，組合せ回路のテスト生成アルゴリズムでテスト生成することはできない [5]．したがって，ここではまず対象回路を無閉路構造の (フィードバックループの無い) 順序回路に限定する．また，話を簡単にするためフリップフロップ (以下，FF と略す) は DFF に限定する．以下，本論文では対象とする故障モデルは信号線の縮退故障とする．

回路中の分岐点において，分岐点の入力側の信号線を**分岐幹**，分岐点の出力側の複数の信号線を**分岐枝**と呼ぶ．経路上に含まれる FF の個数をその経路の**順序深度**という．順序回路の外部入力から外部出力に至る経路の中で最大の順序深度を**順序回路の順序深度**とする．x を外部入力，x_i と x_j を x の分岐枝とするとき，x_i と x_j から等しい順序深度で同じ外部出力 z_k に至る経路が存在しないならば x_i と x_j は**分離可能**という．

集合 X を互いに素な部分集合 $X_1, X_2, \ldots, X_n$ に分け，$X = \bigcup_{i=1}^{n} X_i$ となると

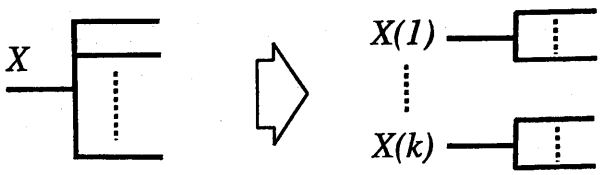

図 2.1 外部入力分離

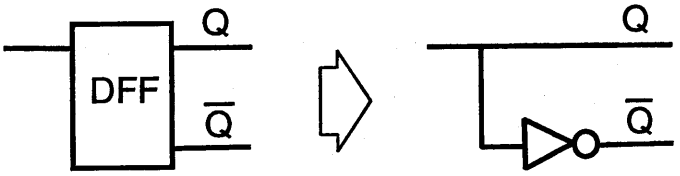

図 2.2 FF の消去

き，これらの部分集合の 集合 $\{X_1, X_2, \ldots, X_n\}$ を集合 X の**分割**といい，各部分集合 X_i $(i = 1, 2, \ldots, n)$ を**ブロック**という．二つの分割 π_1, π_2 に対して，もし π_1 の各ブロックが π_2 のブロックに含まれるとき，$\pi_1 \leq \pi_2$ と書く．ある条件 C に対して，C を満たす X の**極小分割** π とは，条件 C を満たし $\pi' \leq \pi$ となる π' が存在しない分割をいう．

組合せ変換 (C 変換)[8]：

無閉路構造の順序回路 S に対するつぎの 2 つの操作による変換を**組合せ変換 (C 変換)** と呼び，変換されてできる組合せ回路を $C(S)$ と書く．

(1) 分岐枝を有する外部入力について，その外部入力の分岐枝の集合を X とする．「分岐枝 x_i と x_j が分割 π の異なるブロック $X(i), X(j)$ に属する $(x_i \in X(i), x_j \in X(j); X(i) \neq X(j))$ ならば x_i と x_j は分離可能である」を満たす X の極小分割 π を求める．分割した各ブロック毎に新たに外部入力を設けて，もとの外部入力を分離する (図 2.1 参照)．

(注：外部入力を分離する場合，分岐幹の故障はその分岐枝すべてに同時に存在する多重故障として扱う)

(2) FF を信号線に置き換える (FF の否定出力の場合は，NOT ゲートを付加する．図 2.2 参照)． □

組合せテスト生成複雑度でのテスト生成可能性 [8]：

「S を無閉路順序回路，$C(S)$ をその C 変換された組合せ回路とする．S にお

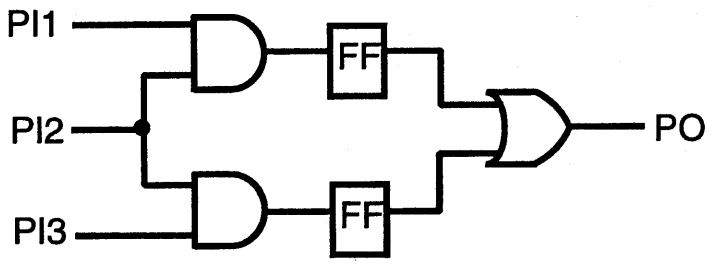

図 2.3 平衡構造の例

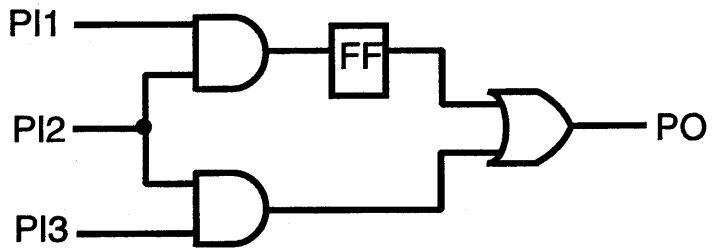

図 2.4 内部平衡構造の例

ける故障 f が S でテスト可能であるための必要十分条件が $C(S)$ における f に対応する故障 f_c が $C(S)$ においてテスト可能であることである」がいえるならば，順序回路 S は，**組合せテスト生成複雑度でテスト生成可能である** (以下，**組合せテスト生成可能であると略す**) という．

組合せテスト生成可能な順序回路としては以下のものがある．

平衡構造 [7]： 回路 S の任意の外部入力と外部出力の対について，その 2 点間のどの経路の順序深度も等しければ，S は**平衡構造**であるという (図 2.3 参照)．

内部平衡構造 [8]： 回路 S に C 変換の操作 (1) を施してできる回路 S' が平衡構造となるならば，S は**内部平衡構造**であるという (図 2.4 参照)．

平衡構造の順序回路ではすべての外部入力が分離不可能であるので，C 変換は操作 (2) だけで行われることになる．したがって定義より，内部平衡構造は平衡構造より広い組合せテスト生成可能な順序回路のクラスと言える．

2.3. 内部平衡部分スキャン設計

次にフィードバックループ (閉路) のある一般の順序回路について考える．このとき，スキャン FF を除いた残りの回路 (核回路) が内部平衡構造となるように部分スキャン設計を行うことを**内部平衡部分スキャン設計**と呼ぶ．内部平衡部分スキャン設計を行えば，組合せ回路のテスト生成アルゴリズムだけでテスト生成を行うことができる．これについては 2.7節で示す．さらに，2.8節でベンチマーク回路に対する実験結果より内部平衡部分スキャン設計が少ない面積オーバヘッドで実現できることを示す．

2.4. 拡張部分スキャン設計

順序回路 S に対して，S の中に含まれる要素 e (FF，信号線) を**取り除く**というのは，e への入力を S の外部出力，e からの出力を S の外部入力に置き換え，要素 e を S から切り離す操作のことを言う．このとき，新たに置き換えられたこれらの外部入力，外部出力のことをそれぞれ**疑似入力**，**疑似出力**と呼ぶ．さらにこの操作を演算子 $-$ で表し，S に含まれる要素の集合 E に対し，$S-E$ は E に属するすべての要素を S から取り除いた回路を表すものとする．

順序回路 S があるとき，その中の FF，信号線の集合をそれぞれ F, L とする．S に含まれる要素の集合 $E = \{FF_1, \ldots, FF_n, l_1, \ldots, l_m\}$ (ただし，$FF_i \in F$ $(i = 1, \ldots, n)$, $l_j \in L$ $(j = 1, \ldots, m)$) に対して，$S-E$ を**核回路** (kernel circuit)，E に属する FF，信号線をそれぞれ**外部 FF** (external flip-flops)，**外部信号線** (external wires) と呼ぶ (図 2.5 参照)．

これまでほとんどの部分スキャン設計では，外部 FF をスキャン **FF** に置き換えて，スキャン FF を等価的に外部入出力とみなすことにより，スキャン FF を取り除いた回路 (核回路) に対し，テスト生成が行われてきた．一方，FF ばかりでなく信号線をスキャン FF と同等の働きをする**バイパス FF** に置き換える方法も提案されている [2, 11, 13]．図 2.6 のように，外部 FF をスキャン FF に，外部信号線をバイパス FF に置き換える部分スキャン設計を**拡張部分スキャン設計**と言う．また，拡張部分スキャン設計された回路において，スキャン FF，バイパ

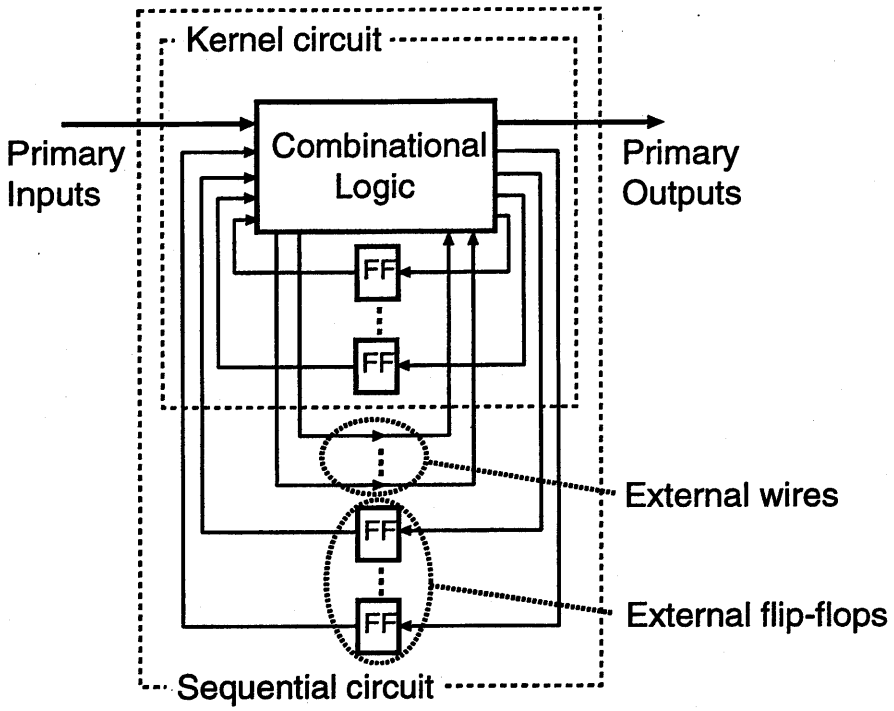

図 2.5 核回路

スFFをまとめて**核外FF**と言う.

スキャンFF, バイパスFF(以下, それぞれS-FF, B-FFと記す)としては, 図2.7, 2.8に示すような構成を考える. 各S-FF, B-FFは表2.1に示すようにマルチプレクサの制御線の値により, **通常動作モード**, **ロードモード**, **シフトモード**の3つの動作モードを設定する. また, FFに与えるクロックにより, **ホールドモード**を設定する.

通常動作モードでは, 図2.6 (a)のようにS-FFはDFFとして, B-FFは信号線として機能する. **ロードモード**では, 核回路からの出力がS-FF, B-FF内のFFに取り込まれ, そのFFの値がS-FF, B-FFの値として出力される. **シフトモード**では, スキャンパスを通してテストパターンがS-FF, B-FF内のFFにスキャンインされ, 同時にそれらFFの値がスキャンアウトされる. FFの**ホールド**とは, データの値を連続したクロックサイクルの間, 保持する機能のことを言う. また, このときのFFの動作モードのことを**ホールドモード**と呼ぶ. 拡張部分スキャン設計における各核外FFは, ホールド機能を備えているものとする.

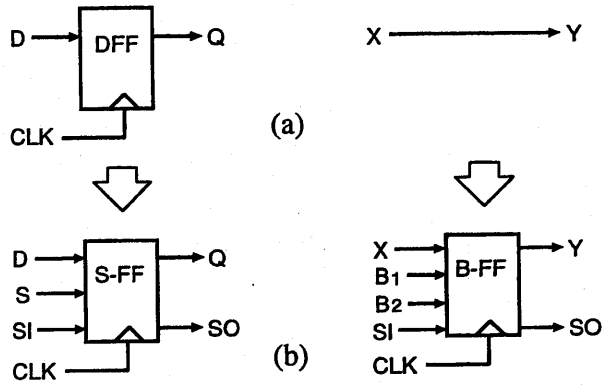

図 2.6 拡張部分スキャン設計

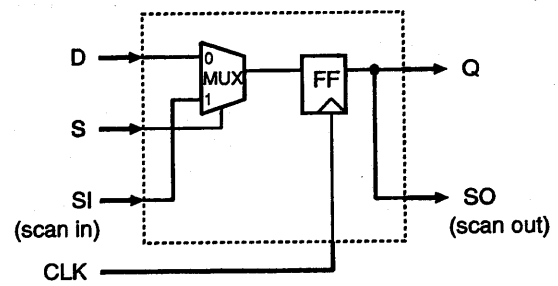

図 2.7 スキャン FF の構成

以上のように，拡張部分スキャン設計では完全スキャン設計と比較して B-FF に含まれるマルチプレクサの制御 (表 2.1 の B_1, B_2入力) および FF のホールド機能の制御に余分な外部入力が必要になる．すなわち，B-FF を用いないときの付加入力数は (完全スキャン設計回路の付加入力数)+1 で，B-FF を用いるときは表 2.1 の通常動作モード時の X の割当てに依存して +2 または +3 となる．

FF を 1 つの S-FF に置き換えることによって増加するハードウェア量を k_1，信号線を 1 つの B-FF に置き換えることによって増加するハードウェア量を k_2 とすると，拡張部分スキャン設計における回路全体の**面積オーバーヘッド** (ハードウェア増加量) は以下の式で表現できる．

回路全体の面積オーバーヘッド $= k_1 \times$ (外部 FF 数) $+ k_2 \times$ (外部信号線数)

k_1, k_2 の具体的な値としては，例えば，文献 [12] のようにマルチプレクサと FF のハードウェア量をそれぞれ 3 と 6 とすれば，k_1 と k_2 の比は 3 : 12 となる．

拡張部分スキャン設計における面積オーバヘッドとして，このほかにも核外 FF

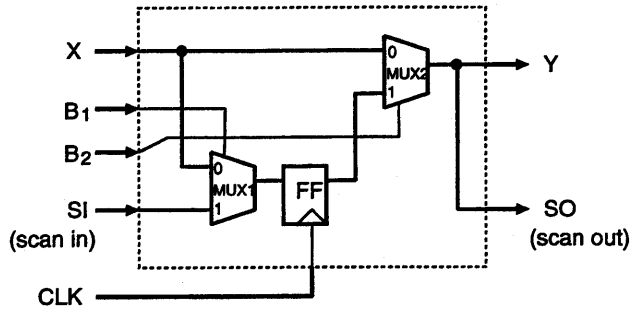

図 2.8 バイパス FF の構成

表 2.1 各動作モードと制御線

	S-FF	B-FF	
	S	B_1	B_2
通常動作モード	0	X	0
ロードモード	0	0	1
シフトモード	1	1	1

にホールド機能を持たせるための面積オーバヘッドが考えられる．ただし，この機能は核外 FF のクロックとそれ以外の FF のクロックを別系統にすることによって実現することができ，その場合は FF のホールド機能の面積オーバーヘッドを無視することができる．したがって，ここでは核外 FF のホールド機能付加による面積オーバーヘッドを評価の対象外としている．

拡張部分スキャン設計は，FF だけをスキャン化する部分スキャン設計よりも信号線を選べる分だけ選択の自由度が大きくなり，回路全体の面積オーバーヘッドを小さくすることが期待できる．

2.5. 内部平衡拡張部分スキャン設計

一般の順序回路について，核回路が内部平衡構造となるように拡張部分スキャン設計を行えば，内部平衡部分スキャン設計の場合と同じように，組合せ回路の

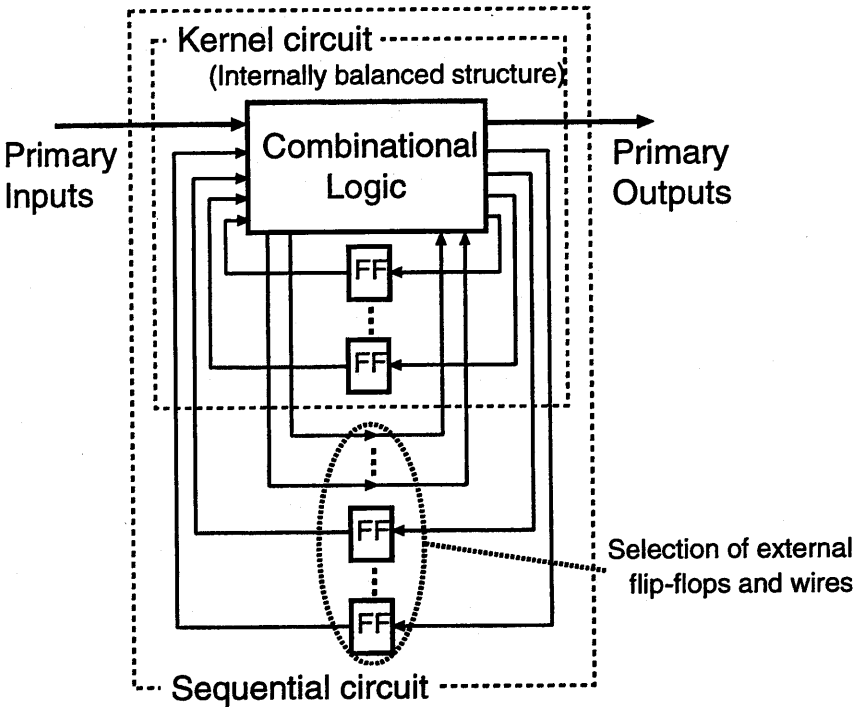

図 2.9 拡張スキャン FF 選択問題

テスト生成アルゴリズムだけでテスト生成を行うことができる．これについては2.7節で示す．

核回路を内部平衡構造とする拡張部分スキャン設計をここでは**内部平衡拡張部分スキャン設計**とよぶ．内部平衡拡張部分スキャン設計において，スキャン化による面積オーバーヘッドを小さくする外部 FF と外部信号線を選択する方法について次の 2.6節で示す．

2.6. 拡張スキャン FF 選択問題

図 2.9 のように任意に与えられた順序回路から核回路が内部平衡構造となるときの面積オーバーヘッドを小さくする外部 FF および外部信号線の集合を求める問題を**拡張スキャン FF 選択問題**と言う．

この拡張スキャン FF 選択問題を，次の 2 段階に分けて解く方法を考える．2 段階に分けることにより，個々の段階で最適解が得られてもそれらは必ずしも全

体の最適解となるわけではないが，問題を細分化することにより，より解きやすい形にすることができる上，既存のアルゴリズムに変更を加えるだけで利用できるという利点がある．

1. 無閉路構造

核回路が無閉路構造となるように面積オーバーヘッド最小の外部 FF および外部信号線を選択する．

2. 内部平衡構造

1 で得られた無閉路構造の核回路に対して，

(2-1) 分離可能な外部入力枝を分離する (核回路の疑似入力は対象外)．

(2-2) (2-1) で得られた回路について，平衡構造にするための面積オーバーヘッド最小の外部 FF および外部信号線を選択する．

これらの問題を解くために，順序回路を次のようなグラフに置き換えて考える．

定義 2.1 順序回路 S の**回路トポロジーグラフ (CTG)** は，次のような重みつき有向グラフ $G=(V,A,w)$ である．

V は S のゲート，分岐点，外部入力，外部出力を頂点とする集合．

$A \subset V \times V$ は S の FF，信号線を辺とする集合 (各辺は順序回路 S における接続関係を表す)．

$w : A \to Z^+$ (正の整数) は辺の重み (FF に対して k_1，信号線に対して k_2．ここで k_1 は FF を 1 つの S-FF に置き換えるときのハードウェア増加量で，k_2 は信号線を 1 つの B-FF に置き換えるときのハードウェア増加量である)．
□

CTG G は以下の変換規則を適用して，より頂点数の少ないグラフ $\hat{G}$ に変換することができる．この変換規則から明らかなように，最初のグラフ G と変換後のグラフ $\hat{G}$ では，オーバーヘッドの等しい拡張スキャン FF 選択問題の解が得られる．

変換規則： CTG G において，外部入出力以外の頂点とその中のすべての辺の重みが k_2 (信号線) だけからなる G の無閉路部分グラフを G_v とする．G_v に入ってくる辺の重み和を w_i，G_v から出ていく辺の重み和を w_o とする．さらに，G_v に入って G_v から出ていくすべての有向道をカットするのに必要な辺の最小重み

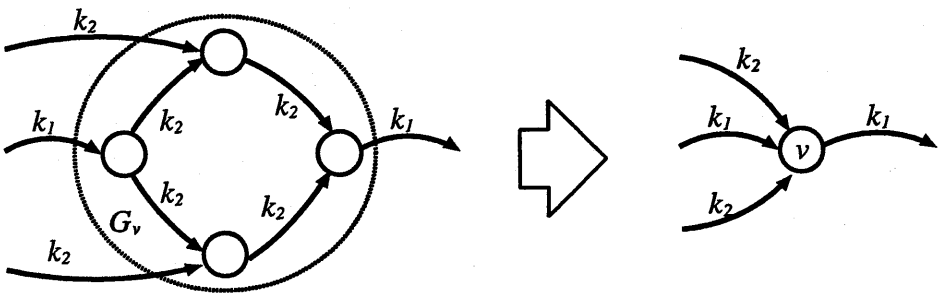

図 2.10 グラフの変換規則

和を m_v とする．このとき，G_v の各入力から G_v のすべての出力へパスがあり，かつ，

$$m_v \geq min\{w_i, w_o\}$$

ならば，G_v を図 2.10 のように 1 つの頂点 v にマージする． □

以上のグラフによる表現を用いれば，核回路が内部平衡構造となる外部 FF および外部信号線を選択する手続きは，以下のように記述できる．

Step 1. 順序回路 S の CTG G に変換規則を適用して，$\widehat{G}$ を求める．この $\widehat{G}$ を $\widehat{G} = (V, A, w)$ とする．

Step 2. $\widehat{G}$ から $\sum_{a \in R_A} w(a)$ が最小となる辺集合 R_A を除去することにより，無閉路グラフ G_A を求める．

Step 3. 無閉路グラフ G_A から分離可能な外部入力を分離することにより，$\widetilde{G}_A$ を求める．

Step 4. $\widetilde{G}_A$ から $\sum_{a \in R_B} w(a)$ が最小となる辺集合 R_B を除去することにより，平衡グラフ G_B を求める． □

以上の結果，$R = R_A \cup R_B$ が求める外部 FF と外部信号線の集合となる．

上記の Step 1 の変換規則で重み和 m_v は最小でなければならないので，無閉路部分グラフ G_v はサイズの小さいものだけに限定する．さらに，Step 2 および Step 4 における最小化問題は NP 完全であることが知られている [7] ので，発見的手法を考える必要がある．

Step 2 において $\sum_{a \in R_A} w(a)$ が最小となる辺集合 R_A を求めるには，$\widehat{G}$ に対して，重みつき MFAS (最小フィードバック辺集合) アルゴリズムを適用させればよい．ここで，各辺の重みは FF が k_1，信号線が k_2 とする．

Step 3 における外部入力分離の操作は C 変換の操作 (1) に該当し，次のような手順で行われる．すなわち，各外部入力 x について，そのすべての分岐枝の組 (x_i, x_j) に対して等しい順序深度で同じ外部出力 z_k に至る経路が存在するかどうかを調べ，存在すれば (x_i と x_j は分離不可能) x_i と x_j を同じ集合の中に入れ，存在しなければ，それぞれ別々の集合を設けてその中に入れる．その結果，生成された集合ごとに新たに外部入力を設ける．

Step 4 において $\sum_{a \in R_B} w(a)$ が最小となる辺集合 R_B は，文献 [7] の balancing procedure を変更した手続きで求めることができる．文献 [7] の balancing procedure では各辺の重みがレジスタのビット幅となっていたが，ここでは辺の重みを FF は k_1，信号線は k_2 に置き換えて考えればよい．

2.7. スキャン設計回路のテスト法

内部平衡部分スキャン設計および内部平衡拡張部分スキャン設計された回路のテストの手続きとして，スキャンパスを用いた核回路に対するテストとスキャン化により新たに付加した回路のテストの 2 つが考えられる．本節ではこれらのテスト生成法について考察し，これらのテスト生成法を用いて生成されるテスト系列が正しいテスト系列であることの正当性を示す．以下の議論は内部平衡拡張部分スキャン設計された回路について述べているが，B-FF が存在しない場合を考えれば，内部平衡部分スキャン設計された回路にもそのまま適用することができる．

順序回路を S，S に対して内部平衡拡張部分スキャン設計を行った回路を S_{DFT} とする．また，S_{DFT} において，核外 FF を取り除いた回路 (内部平衡構造となる核回路) を S_K とする．また，S_K を C 変換した回路を $C(S_K)$ とする．

順序回路 S に対し，S_{DFT} のように拡張部分スキャン設計を行ってテストをするためには，S_{DFT} のすべての故障に対してテストが行われなければならない．このうち，S_{DFT} において S_K に対応する故障のことを S_{DFT} の**核内の故障**と呼ぶ．一方，S_{DFT} にあって，S_K にない故障 (S_{DFT} の核外 FF の故障) のことを S_{DFT} の**核外の故障**と呼ぶ．それぞれの故障に対するテストについて，以下に述べる．

2.7.1 核内の故障に対するテスト

S_{DFT} の核内の故障を f_{DFT} とする．f_{DFT} に対応する S_K の故障を f_K とする．さらに f_K に対応する $C(S_K)$ の故障を f_C とする．f_{DFT}，f_K，f_C はそれぞれ一対一に対応している．

核回路 S_K が内部平衡構造となるように拡張部分スキャン設計を行った回路 S_{DFT} に対して，S_{DFT} の核内の故障 f_{DFT} のテスト系列は以下の手順で求められる．

1. f_{DFT} に対応する $C(S_K)$ の f_C について，組合せ回路のテスト生成アルゴリズムを使ってテスト生成を行う．
2. 生成されたテストパターンを S_K のテスト系列に変換する．
3. S_K のテスト系列を S_{DFT} のテスト系列に変換する．

1 ～ 3 の手順については，後で詳述する．この手順では，S_{DFT} での f_{DFT} のテスト生成問題は，$C(S_K)$ での f_C のテスト生成問題に帰着している．これでテスト生成できることを保証するには，「S_{DFT} における故障 f_{DFT} が S_{DFT} でテスト可能であるための必要十分条件は，$C(S_K)$ における f_{DFT} に対応する故障 f_C が $C(S_K)$ においてテスト可能であることである」ことを示す必要がある．これを以下の定理で示す．

文献 [8] より，順序回路 S_K が内部平衡構造ならば，S_K は組合せテスト生成可能であるので，次の定理が成立する．

定理 2.1 順序回路 S_K は内部平衡構造であるとする．S_K における故障 f_K が S_K でテスト可能であるための必要十分条件は，$C(S_K)$ における f_K に対応する故障 f_C が $C(S_K)$ においてテスト可能であることである． □

定理2.1をもとに，S_K が内部平衡構造の場合，組合せ回路 $C(S_K)$ でテスト生成が行われ，それが S_K，S_{DFT} のテスト系列に変換される手順を以下に示す．

手順1： $C(S_K)$ におけるテスト生成　まず，S_K をC変換して $C(S_K)$ を求める．$C(S_K)$ は組合せ回路なので，f_K に対応する故障 f_C が $C(S_K)$ でテスト可能であれば，組合せ回路のテスト生成アルゴリズムを用いて，f_C に対するテストパターンが求められる（f_K が S_K で冗長であれば，f_C は $C(S_K)$ で冗長である

表 2.2 テストパターン T_C とテスト系列 T_K

(a) T_C

入力	値
x_1	a_1
⋮	⋮
x_i	a_i
⋮	⋮
x_{h1}	a_{h1}
x_{h2}	a_{h2}
⋮	⋮
x_m	a_m
y_1	b_1
⋮	⋮
y_p	b_p
⋮	⋮
y_l	b_l

(b) T_K

	時刻 t										
入力	1	⋯	$d+1-d_{ik}$	⋯	$d+1-d_{h1k}$	⋯	$d+1-d_{h2k}$	⋯	d	$d+1$
x_1		⋯		⋯		⋯		⋯		
⋮										
x_i	×	⋯	a_i	⋯	×	⋯	×	⋯	×	×
⋮										
x_h	×	⋯	×	⋯	a_{h1}	⋯	a_{h2}	⋯	×	×
⋮										
x_m		⋯		⋯		⋯		⋯		
y_1	b_1	⋯	b_1	⋯	b_1	⋯	b_1	⋯	b_1	b_1
⋮	⋮		⋮		⋮		⋮		⋮	⋮
y_p	b_p	⋯	b_p	⋯	b_p	⋯	b_p	⋯	b_p	b_p
⋮	⋮		⋮		⋮		⋮		⋮	⋮
y_l	b_l	⋯	b_l	⋯	b_l	⋯	b_l	⋯	b_l	b_l

ので (定理 2.1), f_C に対するテストパターンは存在しない). f_C に対するテストパターンが求められたとき, このテストパターンを T_C とする.

手順 2： S_K におけるテスト系列　S_K の順序深度を d とすると, S_K のテスト系列 T_K の長さは $d+1$ となる. ここで f_C はテストパターン T_C によって $C(S_K)$ の出力 z_k (外部出力もしくは疑似出力) で検出されるものとする. f_C に対応する故障 f_K が時刻 $d+1$ に S_K の出力 z_k で検出されるように, テスト系列 T_K の外部入力 x_i $(i=1,2,\ldots,m)$ および疑似入力 y_p $(p=1,2,\ldots,l)$ の値を以下のように決める (表 2.2 参照).

(1) 外部入力 x_i $(i=1,2,\ldots,m)$ の値の決め方

(1-a) x_i が分離不可能な外部入力の場合：

x_i から出力 z_k への順序深度 d_{ik} は一意的に決まる. テストパターン T_C の外部入力 x_i の値をテスト系列 T_K の時刻 $d+1-d_{ik}$ のときの外部入力 x_i の値と決める.

(1-b) x_i が分離可能な外部入力の場合：

S_K における外部入力 x_i が $C(S_K)$ において外部入力 $x_{i1}, x_{i2}, \ldots, x_{in}$ に分離されているとする. 内部平衡構造であるので, 各 x_{ij} から z_k への順序深度は一意的に決まり, それを d_{ijk} とする. それらは分離可能であるので, d_{ijk} $(j=1,2,\ldots,n)$ はすべて異なる順序深度となる. したがって, 時刻 $d+1-d_{ijk}$ $(j=1,2,\ldots,n)$

はすべて異なり，テスト系列 T_K の n 箇所の時刻に値を設定する．すなわち，テストパターン T_C の各外部入力 x_{ij} $(j=1,2,\ldots,n)$ の値を，テスト系列 T_K の時刻 $d+1-d_{ijk}$ $(j=1,2,\ldots,n)$ の外部入力 x_i の値と決める．

(2) 疑似入力 y_p $(p=1,2,\ldots,l)$ の値の決め方

疑似入力 y_p $(p=1,2,\ldots,l)$ はすべて分離不可能な入力となっている．よって，y_p から出力 z_k への順序深度 d_{ypk} は一意的に決まる．テストパターン T_C の疑似入力 y_p の値をテスト系列 T_K の時刻 $d+1-d_{ypk}$ のときの疑似入力 y_p の値と決める．後で S_{DFT} 対応のテスト系列を作ることを考慮して，y_p の $d+1-d_{ypk}$ 以外の時刻にも時刻 $d+1-d_{ypk}$ と同じ値を割り当てる．したがって，結果的には S_K の疑似入力 y_p $(p=1,2,\ldots,l)$ の値はどの時刻もテストパターン T_C の疑似入力 y_p の値と同じ値になっている．

以上のようにして，時刻 1, 2, ..., d, $d+1$ での外部入力，疑似入力の値が決まる．時刻 t における外部入力ベクトルを X_t，疑似入力ベクトルを Y_t とすると，S_K の故障 f_K に対するテスト系列 T_K は外部入力系列 $[X_1,X_2,\ldots,X_d,X_{d+1}]$，疑似入力系列 $[Y_1,Y_2,\ldots,Y_d,Y_{d+1}]$ $(Y_1=Y_2=\cdots=Y_d=Y_{d+1})$ となる．

手順3： S_{DFT} におけるテスト系列　S_K では対応する時刻ごとに各入力ベクトルを割り当てることにより，$d+1$ 時刻目の応答を観測して，テストを行っている．S_{DFT} では S_K における疑似入出力が S-FF，B-FF に置き換わっているので，疑似入力がスキャンパスを通しての入力に変わっていることを考慮して，テストを行うようにする．

S_K の故障 f_K に対応する S_{DFT} の故障 f_{DFT} のテスト系列 T_{DFT} は以下の手順となる．

1. すべての核外 FF をシフトモードにして，疑似入力ベクトル Y_1 をスキャンインする．

2.(a) すべての核外 FF をホールドモード，すべての内部 FF をロードモードにして，外部入力系列 $X_1,X_2,\ldots,X_d$ を印加する．

(b) すべての核外 FF をロードモードにして，X_{d+1} を入力．

(c) すべての核外 FF をシフトモードにして，スキャンアウト．

上に示した手順は S_{DFT} のある故障 f_{DFT} に対するものであったが，S_{DFT} の検出可能なすべての故障に対してテストを行う際には，上記の 2. の操作はそのテ

スト数分だけ繰り返される (この場合，2.(c) のところでスキャンアウトすると同時に次のテストパターンをスキャンインする)．

以上のようにして，S_K が内部平衡構造の場合，組合せ回路 $C(S_K)$ でテスト生成が行われ，S_{DFT} のテスト系列に変換される．

以上のテスト系列の変換手順 1 ～ 3 と定理 2.1から次の定理が成立する．

定理 2.2 順序回路 S_K は内部平衡構造であるとする．S_K における故障 f_K が冗長故障ならば，f_K に対応する $C(S_K)$ の故障 f_C が手順1において冗長故障と判定される．S_K における故障 f_K がテスト可能ならば，手順2, 3で得られる系列 T_K，T_{DFT} は，各々 S_K，S_{DFT} において対応する故障 f_K，f_{DFT} のテスト系列になっている． □

定理 2.2が S_{DFT} の核内の故障についても成立することを言うには，次の定理を示す必要がある．

定理 2.3 順序回路 S_K は内部平衡構造で，S_{DFT} の核外 FF はホールド機能を持つとする．このとき，S_K における故障 f_K が S_K でテスト可能であるための必要十分条件は，S_{DFT} における f_K に対応する故障 f_{DFT} (S_{DFT} の核内の故障) が S_{DFT} においてテスト可能であることである．

(証明) 必要条件の証明は上に述べたテスト系列の変換手順から明らかである．十分条件については S_{DFT} を核回路とそれ以外の回路に分けて考えれば，S_{DFT} の核回路の故障 f_{DFT} に対するテストが S_K の故障 f_K に対するテストになることを簡単に示すことができる． □

2.7.2 核外の故障に対するテスト

次に，S_{DFT} の核外の故障 (核外 FF の故障) をテストする方法について考える．

ここで対象としている核外の故障は，S-FF，B-FF 内部の信号線およびマルチプレクサの故障で，これは図 2.11，2.12 の S-FF と B-FF においてアルファベットで記した信号線の 0/1 縮退故障である．

図 2.11，2.12 で，S-FF の D，B-FF の X をデータ入力，S-FF の S，B-FF の B_1, B_2 を制御入力，S-FF，B-FF の SI をスキャン入力と呼ぶ．また，各 S-FF および B-FF にスキャンデータを入れるための入力をスキャンパスの入力と呼ぶ．

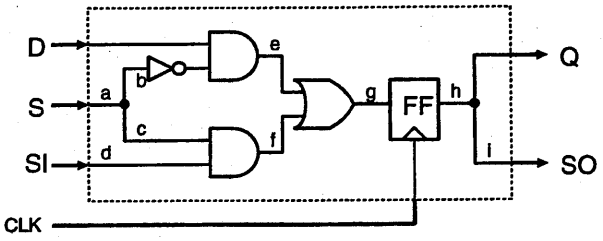

図 2.11 S-FF

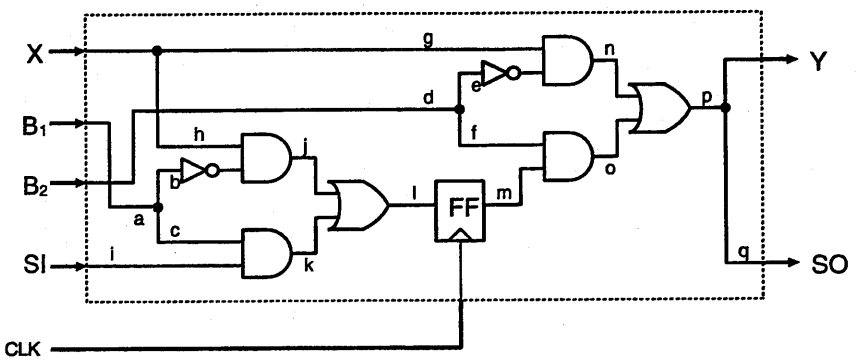

図 2.12 B-FF

また，各核外 FF (S-FF，B-FF) FF_i $(i = 1, 2, \ldots, n;\ n$：核外 FF の個数) に対し，図 2.13 のように FF のデータ入力に接続されている信号線を FO_i，図 2.11 の h，図 2.12 の p に相当する信号線を FI_i とする．

図 2.11，2.12 に対して，各信号線の故障を検出するための S-FF，B-FF のテストパターンはそれぞれ表 2.3，表 2.4 のようになる．また，その中に示されているように，対応するテストパターンにより，故障集合を A ～ G の 7 つのクラスに分ける．

クラス A ～ D の故障集合は，S-FF と B-FF に共通なものである．それぞれ，クラス A のテストパターンは S-FF(B-FF) においてデータ入力 $D(X)$ がドント・ケアであるもの，クラス B のテストパターンは S-FF(B-FF) においてスキャン入力 SI がドント・ケアであるもの，クラス C, D のテストパターンは S-FF(B-FF) においてデータ入力 $D(X)$，スキャン入力 SI，制御入力 S (B_1, B_2) のいずれも 0/1 の値が入るものとなっている．

また，クラス E ～ G の故障集合は，B-FF に特有なものである．いずれもマ

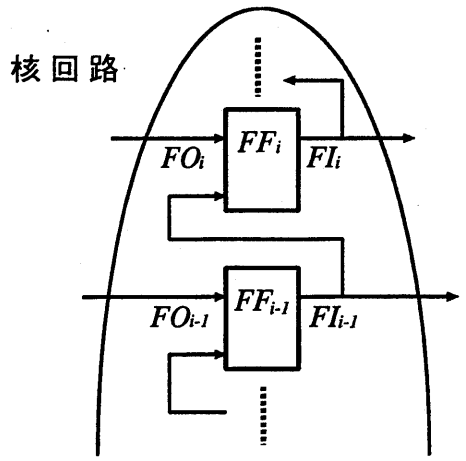

図 2.13 核外 FF

ルチプレクサ MUX2 (図 2.8 参照) の入力に関わるもので，クラス E のテストパターンは MUX2 において，FF からの入力がドント・ケアであるもので，クラス F, G のテストパターンは MUX2 において，データ入力 X，FF からの入力，制御入力 B_1, B_2のいずれも 0/1 の値が入るものとなっている (FF からの入力に 0/1 のパターンを与えるために，ここでは前の時刻にスキャン入力から 0/1 のパターンを入れている).

さらに，S_{DFT} の核外の故障に対するテストを考える上で，次のようなテスト系列について定義する.

テスト系列 α： すべての S-FF，B-FF をシフトモードにして，スキャンパスの入力から (スキャンパス長+1) クロックだけ 0, 1 のパターンを交互に入れるテスト系列.

テスト系列 β_i^v $(i = 1, 2, \ldots, n,\ v = 0/1)$： 核外 FF FF_i に接続されている信号線 FO_i の非冗長故障 FO_i/v を検出するテスト系列.

テスト系列 γ_i^v $(i = 1, 2, \ldots, n,\ v = 0/1)$： 核外 FF FF_i に対して以下のような手続きで行うテスト系列

1. S_{DFT} の内部平衡構造となる核回路 S_K において，$FO_i = v$, $FI_{i-1} = \bar{v}$ となる系列を求める (これは S_K を C 変換した回路 $C(S_K)$ で $FI_{i-1} = \bar{v}$

表 2.3 S-FF の 2 テストパターンと検出される故障

クラス		現在の時刻			次の時刻			テストされる故障	
		D	S	SI	D	S	SI	現時刻	次時刻
A	A_0	X	1	0	X	1	X	d/1, f/1, g/1	h/1, i/1
	A_1	X	1	1	X	1	X	c/0, d/0, f/0, g/0	h/0, i/0
B	B_0	0	0	X	X	1	X	e/1	
	B_1	1	0	X	X	1	X	b/1, e/0	
C	C_0	0	0	1	X	1	X	a/1, c/1	
	C_1	1	0	0	X	1	X	a/1	
D	D_0	0	1	1	X	1	X	a/0	
	D_1	1	1	0	X	1	X	a/0, b/0	

表 2.4 B-FF の 2 テストパターンと検出される故障

クラス		現在の時刻				次の時刻				テストされる故障	
		X	B_1	B_2	SI	X	B_1	B_2	SI	現時刻	次時刻
A	A_0	X	1	1	0	X	1	1	X	i/1, k/1, l/1	m/1, o/1, p/1, q/1
	A_1	X	1	1	1	X	1	1	X	c/0, i/0, k/0, l/0	f/0, m/0, o/0, p/0, q/0
B	B_0	0	0	X	X	X	1	1	X	h/1, j/1	
	B_1	1	0	X	X	X	1	1	X	b/1, h/0, j/0	
C	C_0	0	0	1	1	X	1	1	X	a/1, c/1	
	C_1	1	0	1	0	X	1	1	X	a/1	
D	D_0	0	1	1	1	X	1	1	X	a/0	
	D_1	1	1	1	0	X	1	1	X	a/0, b/0	
E	E_0	0	1	0	X	X	1	1	X	g/1, n/1	
	E_1	1	1	0	X	X	1	1	X	e/1, g/0, n/0	
F	F_0	X	1	1	1	0	1	0	X		d/1, f/1
	F_1	X	1	1	0	1	1	0	X		d/1
G	G_0	X	1	1	1	0	1	1	X		d/0
	G_1	X	1	1	0	1	1	1	X		d/0, e/0

の条件のもとで $FO_i/\bar{v}$ についてテスト生成を行うことによって求められる). S_K でこの系列が存在すれば，時刻 1, 2, ..., d, $d+1$ の外部入力，疑似入力のベクトルをそれぞれ **PI** $[X_1, X_2, \ldots, X_d, X_{d+1}]$, **SI** $[Y_1, Y_1, \ldots, Y_1, Y_1]$ とする (d は S_K の順序深度．以下のテスト系列の場合も同様に定義する)．そして，S_{DFT} において以下の手続きを実行する．

2. すべての核外 FF をシフトモードにして，Y_1 をスキャンイン．
3. (a) すべての核外 FF をホールドモード，すべての内部 FF をロード

モードにして，$X_1, X_2, \ldots, X_d$ を印加する．

(b) 核外 FF をロードモードにして，X_{d+1} を入力．

(c) すべての核外 FF をシフトモードにして，FF_i の中身をスキャンアウト．

テスト系列 δ_i^v $(i = 1, 2, \ldots, n,\ v = 0/1)$： 核外 FF FF_i に対して以下のような手続きで行うテスト系列

1. テスト系列 γ_i^v の 1. と同様．$FO_i = v$, $FI_{i-1} = \bar{v}$ となる S_K の系列が存在すれば，S_{DFT} において以下の手続きを実行する．
2. すべての核外 FF をシフトモードにして，Y_1 を 3 回スキャンイン．2 回続けて同じ Y_1 がスキャンアウトされれば，以下の手続きを実行する．
3. (a) テスト系列 γ_i^v と同様．

   (b) 核外 FF をシフトモードにして，X_{d+1} を入力．

   (c) テスト系列 γ_i^v と同様．

テスト系列 ϵ_i^v $(i = 1, 2, \ldots, n,\ v = 0/1)$： B-FF FF_i に対して以下のような手続きで行うテスト系列

1. S_{DFT} の内部平衡構造となる核回路 S_K において，$FO_i = v$ となる系列を求める (これは S_K を C 変換した回路 $C(S_K)$ で $FO_i/\bar{v}$ についてテスト生成を行うことによって求められる)．S_K でこの系列が存在すれば，S_{DFT} において以下の手続きを実行する．
2. テスト系列 γ_i^v と同様．
3. (a) テスト系列 γ_i^v と同様．

   (b) S-FF をシフトモード $(S = 1)$，B-FF で $B_1 = 1$, $B_2 = 0$ にして，X_{d+1} を入力．

   (c) すべての核外 FF をシフトモードにして，FF_{i+1} の中身をスキャンアウト．

テスト系列 ζ_i^v $(i = 1, 2, \ldots, n,\ v = 0/1)$： B-FF FF_i に対して以下のような手続きで行うテスト系列

1. S_{DFT} の内部平衡構造となる核回路 S_K において，$FO_i = v$, $FI_i = \bar{v}$ となる系列を求める (これは S_K を C 変換した回路 $C(S_K)$ で $FI_i = \bar{v}$ の条件のもとで $FO_i/\bar{v}$ についてテスト生成を行うことによって求められる)．S_K でこの系列が存在すれば，S_{DFT} において以下の手続きを実行する．

2. 3. テスト系列 ϵ_i^v と同様．

テスト系列 η_i^v $(i = 1, 2, \ldots, n,\ v = 0/1)$： B-FF FF_i に対して以下のような手続きで行うテスト系列

1. テスト系列 ζ_i^v の 1. と同様．$FO_i = v$, $FI_i = \bar{v}$ となる S_K の系列が存在すれば，S_{DFT} において以下の手続きを実行する．

2. テスト系列 δ_i^v と同様．

3. (a) (b) テスト系列 δ_i^v と同様．
   (c) テスト系列 ϵ_i^v と同様．

これらのテスト系列により，核外 FF の非冗長故障がテスト可能であることを以下に示す．

補題 1 核外 FF に対するクラス A のすべての故障は，テスト系列 α でテスト可能である．

(証明) テスト系列 α を S_{DFT} に印加すれば，各核外 FF について表 2.3, 表 2.4 の A_0, A_1 の 2 つの系列が入る．クラス A のすべての故障は A_0, A_1 の系列でテスト可能なので，テスト系列 α でテスト可能である． □

核外 FF の大半の故障はスキャンパス上にあり，これらはテスト系列 α でテストすることができる．上のテスト系列で検出できなかった故障については，次の補題 2 ～ 7 を適用する．

補題 2 核外 FF FF_i $(i = 1, 2, \ldots, n)$ に対するクラス B の非冗長故障は，テスト系列 $\beta_i = \{\beta_i^0, \beta_i^1\}$ でテスト可能である．

(証明) 核外 FF FF_i に対し，クラス B の B_0 の故障は $FO_i/1$ と等価であ

る (図 2.13 参照). 同様に，B_1 の故障は $FO_i/0$ と等価である (同図参照). $FO_i/1$ と $FO_i/0$ は冗長でなければそれぞれテスト系列 β_i^1, β_i^0 で検出されるので，FF_i のクラス B の非冗長故障は，テスト系列 β_i でテスト可能である. □

この補題より，クラス B の故障は核内の故障に対するテスト系列でテストできるので，新たにテスト生成を行う必要はない.

補題 3 核外 FF FF_i $(i = 1, 2, \ldots, n)$ に対するクラス C の非冗長故障は，テスト系列$\gamma_i = \{\gamma_i^0, \gamma_i^1\}$ でテスト可能である.

(証明) 各テスト系列 γ_i^v $(v = 0/1)$ が存在するときの S-FF，B-FF のテスト可能な故障は表 2.5 のようになる. それぞれの故障集合を C_i^1, C_i^2 とする. 表 2.5 より，

1. $\gamma_i = \{\gamma_i^0, \gamma_i^1\}$ が存在する場合：
   — $C_i^1 \cup C_i^2$ の故障 (クラス C の故障) は γ_i でテスト可能である
2. γ_i が存在しない場合，すなわち，
   (a) γ_i^0 が存在しない場合：
   — C_i^1 は冗長である
   (b) γ_i^0 および γ_i^1 が存在しない場合：
   — C_i^2 は冗長である

したがって，核外 FF FF_i に対するクラス C の非冗長故障はテスト系列 γ_i でテスト可能である. □

補題 4 核外 FF FF_i $(i = 1, 2, \ldots, n)$ に対するクラス D の非冗長故障は，テスト系列$\delta_i = \{\delta_i^0, \delta_i^1\}$ でテスト可能である.

(証明) 各テスト系列 δ_i^v $(v = 0/1)$ が存在するときの S-FF，B-FF のテスト可能な故障は表 2.6 のようになる. それぞれの故障集合を D_i^1, D_i^2 とする. 表 2.6 より，

1. $\delta_i = \{\delta_i^0, \delta_i^1\}$ が存在する場合：
   — $D_i^1 \cup D_i^2$ の故障 (クラス D の故障) は δ_i でテスト可能である

表 2.5 テスト系列 γ_i でテストできる故障

γ_i	テスト可能な故障		故障集合
	S-FF	B-FF	
γ_i^0	c/1	c/1	C_i^1
$\{\gamma_i^0, \gamma_i^1\}$	a/1	a/1	C_i^2

表 2.6 テスト系列 δ_i でテストできる故障

δ_i	テスト可能な故障		故障集合
	S-FF	B-FF	
δ_i^1	b/0	b/0	D_i^1
$\{\delta_i^0, \delta_i^1\}$	a/0	a/0	D_i^2

2. δ_i が存在しない場合，すなわち，
   (a) δ_i^1 が存在しない場合：
   — D_i^1 は冗長である
   (b) δ_i^0 および δ_i^1 が存在しない場合：
   — D_i^2 は冗長である

したがって，核外 FF FF_i に対するクラス D の非冗長故障はテスト系列 δ_i でテスト可能である． □

補題 4 のクラス D の故障に対するテストでは，テスト系列 δ_i^v の 2. の手続きで同じ Y_1 が 2 回続けてスキャンアウトされなければ，その時点で故障が検出され，テストされたことになる．これは後の補題 7 のクラス G の故障に対するテストでも同様である．

次の補題 5 ～ 7 では，B-FF に特有の故障に対するテストを考えている．B-FF FF_i に対してテストをするとき，これらのテストの結果はスキャンパスで次に接続されている核外 FF FF_{i+1} に取り込まれる．

補題 5 B-FF FF_i $(i = 1, 2, \ldots, n)$ に対するクラス E の非冗長故障は，テスト系列 $\epsilon_i = \{\epsilon_i^0, \epsilon_i^1\}$ でテスト可能である．

(証明)　各テスト系列 ϵ_i^v $(v=0/1)$ が存在するときの B-FF のテスト可能な故障は表 2.7 のようになる．それぞれの故障集合を E_i^1, E_i^2 とする．表 2.7 より，

1. $\epsilon_i = \{\epsilon_i^0, \epsilon_i^1\}$ が存在する場合：
   — $E_i^1 \cup E_i^2$ の故障 (クラス E の故障) は ϵ_i でテスト可能である
2. ϵ_i が存在しない場合，すなわち，
   (a) ϵ_i^0 が存在しない場合：
   — E_i^1 は冗長である
   (b) ϵ_i^1 が存在しない場合：
   — E_i^2 は冗長である

したがって，核外 FF FF_i に対するクラス E の非冗長故障はテスト系列 ϵ_i でテスト可能である．　□

補題 6　B-FF FF_i $(i=1,2,\ldots,n)$ に対するクラス F の非冗長故障は，テスト系列$\zeta_i = \{\zeta_i^0, \zeta_i^1\}$ でテスト可能である．
(証明)　各テスト系列 ζ_i^v $(v=0/1)$ が存在するときの B-FF のテスト可能な故障は表 2.8 のようになる．それぞれの故障集合を F_i^1, F_i^2 とする．表 2.8 より，

1. $\zeta_i = \{\zeta_i^0, \zeta_i^1\}$ が存在する場合：
   — $F_i^1 \cup F_i^2$ の故障 (クラス F の故障) は ζ_i でテスト可能である
2. ζ_i が存在しない場合，すなわち，
   (a) ζ_i^0 が存在しない場合：
   — F_i^1 は冗長である
   (b) ζ_i^0 および ζ_i^1 が存在しない場合：
   — F_i^2 は冗長である

したがって，核外 FF FF_i に対するクラス F の非冗長故障はテスト系列 ζ_i でテスト可能である．　□

表 2.7 テスト系列 ϵ_i でテストできる故障

ϵ_i	テスト可能な故障 B-FF	故障集合
ϵ_i^0	g/1, n/1	E_i^1
ϵ_i^1	e/1, g/0, n/0	E_i^2

表 2.8 テスト系列 ζ_i でテストできる故障

ζ_i	テスト可能な故障 B-FF	故障集合
ζ_i^0	f/1	F_i^1
$\{\zeta_i^0, \zeta_i^1\}$	d/1	F_i^2

補題 7 B-FF FF_i $(i = 1, 2, \ldots, n)$ に対するクラス G の非冗長故障は，テスト系列$\eta_i = \{\eta_i^0, \eta_i^1\}$ でテスト可能である．

(証明)　各テスト系列 η_i^v $(v = 0/1)$ が存在するときの B-FF のテスト可能な故障は表 2.9 のようになる．それぞれの故障集合を G_i^1, G_i^2 とする．表 2.9 より，

1. $\eta_i = \{\eta_i^0, \eta_i^1\}$ が存在する場合：
   — $G_i^1 \cup G_i^2$ の故障 (クラス F の故障) は η_i でテスト可能である
2. η_i が存在しない場合，すなわち，
   (a) η_i^1 が存在しない場合：
   — G_i^1 は冗長である
   (b) η_i^0 および ζ_i^1 が存在しない場合：
   — G_i^2 は冗長である

したがって，核外 FF FF_i に対するクラス G の非冗長故障はテスト系列 η_i でテスト可能である．　□

補題 1 ～ 7 より，次の定理が成立する．

表 2.9 テスト系列 η_i でテストできる故障

η_i	テスト可能な故障 B-FF	故障集合
η_i^1	d/0	G_i^1
$\{\eta_i^0, \zeta_i^1\}$	e/0	G_i^2

定理 2.4 核回路が内部平衡構造となるように拡張部分スキャン設計を行った回路を S_{DFT} とする．このとき，*S-FF*，*B-FF* の非冗長故障はテスト可能である． □

2.8. 実験結果

2.6節で示した手法に基づき，核回路を内部平衡構造とするための外部 FF および外部信号線を求める実験を ISCAS'89 ベンチマーク回路に対して行った．ここで，FF を 1 つの S-FF に置き換えるハードウェア増加量を k_1，信号線を 1 つの B-FF に置き換えるハードウェア増加量を k_2 とすると，$k_1 = 1, k_2 = 4$ として面積オーバーヘッドの計算を行った．

実験結果は表 2.10 [1]に掲げた通りである．ここで,「GGB」は核回路を平衡構造とする部分スキャン設計 [7] のときの面積オーバーヘッド (スキャン FF 数),「手法 1」は内部平衡部分スキャン設計のときの面積オーバーヘッド (同),「手法 2」は内部平衡拡張部分スキャン設計のときの面積オーバーヘッド (FF, Wire はそれぞれそのときの外部 FF, 外部信号線数) を示している．表 2.10 の結果より，s15850, s15850.1，s35932 では外部入力分離の効果により手法 1 が GGB[7] より少ない面積オーバーヘッドで実現できることが示された．また，s13207，s13207.1，s38417 では外部信号線を選択する効果が現われて，手法 2 が最も少ない面積オーバーヘッドで実現できることが示された．それ以外の回路については GGB と同じ結果となったが，全体として見ると，手法 1 は GGB に対して，さらに手法 2 は手

[1]回路名に '.1' が付加されている回路は，もとのベンチマーク回路で外部入力から到達不能なフリップフロップを外部入出力に置き換えた回路である．

表 2.10 実験結果

回路名	回路特性				面積オーバーヘッド				
					GGB[7]	手法 1	手法 2		
	ゲート数	入力数	出力数	FF 数			FF	Wire	合計
s1196	388	14	14	18	16	16	16	0	16
s1238	428	14	14	18	16	16	16	0	16
s1423	490	17	5	74	72	72	72	0	72
s5378	1004	35	49	179	124	124	124	0	124
s9234	2027	19	22	228	210	210	210	0	210
s9234.1	2027	36	39	211	193	193	193	0	193
s13207	2573	31	121	669	471	471	409	10	449
s13207.1	2573	62	152	638	441	441	379	10	419
s15850	3448	14	87	597	555	552	552	0	552
s15850.1	3448	77	150	534	483	480	480	0	480
s38417	8709	28	106	1636	1255	1255	1240	3	1252
s38584	11448	12	278	1452	1445	1445	1445	0	1445
s38584.1	11448	38	304	1426	1419	1419	1419	0	1419
s35932	12204	35	320	1728	1728	1704	1704	0	1704

法 1 に対して，同じかそれよりも良い結果が得られている．

2.9. むすび

組合せテスト生成可能な順序回路として内部平衡構造を考え，核回路を内部平衡構造にする拡張部分スキャン設計の手法を示し，その性質を明らかにした．拡張部分スキャン設計では，従来の部分スキャン設計と比較して，FF だけでなく信号線も選択することによって選択の自由度が大きくなり，回路全体の面積オーバーヘッドを小さくすることが期待できる．また，任意に与えられた順序回路から核回路が内部平衡構造となるときの面積オーバーヘッドを小さくする外部 FF および外部信号線を選択する問題について考察した．さらに，核回路を内部平衡構造とする拡張部分スキャン設計された回路では，組合せ回路のテスト生成アルゴリズムで生成されたパターンを変換した系列によりテストすることができることを導いた．最後に，ISCAS'89 ベンチマーク回路に対する実験結果より，提案

した拡張部分スキャン設計が小さい面積オーバヘッドで実現できることを示した.

第 3 章

無閉路部分スキャン設計に基づくテスト容易化高位合成

3.1. はじめに

近年の VLSI の高集積化，大規模化に伴い，回路のテストはますます重要でかつ困難な問題となっている [1]．テストの費用を削減するために，設計の初期の段階からテスト容易性を考慮することが必要とされている．抽象度の高い動作記述からレジスタ転送レベル (RTL) の回路を合成する高位合成の段階でテスト容易性を考慮することにより，回路の面積・性能とともにテスト容易性も含めた最適化および設計費用の削減ができるものと期待されている．本章では，テスト容易性を考慮した高位合成 (テスト容易化高位合成) の一手法として，無閉路構造に基づく部分スキャン設計のためのデータパスのテスト容易化高位合成法を考察する．

一部のフリップフロップをスキャン可能なフリップフロップ (スキャンフリップフロップ) に置き換える部分スキャン設計は，小さいハードウェアオーバヘッドでテスト容易な回路を実現するための重要な技術の一つである．用いるテスト生成アルゴリズムによって，部分スキャン設計法は大きく二つの手法に分けられる．一つは順序回路用テスト生成アルゴリズムを用いることを前提とした部分スキャン設計法で，文献 [5, 6] ではスキャンフリップフロップによってセルフループを除いたフィードバックループを切断する手法が提案されている．もう一つは

組合せ回路用テスト生成アルゴリズムを用いることを前提とした部分スキャン設計法である [7, 8, 9, 10]. この部分スキャン設計法において共通することは，RTL回路の一部のレジスタ (フリップフロップの組) をスキャンレジスタに置き換え，スキャンレジスタによってセルフループを含むすべてのフィードバックループを切断することにより，無閉路構造を実現する**無閉路部分スキャン設計**である．本章での高位合成法は，無閉路部分スキャン設計のスキャンレジスタ数が最小になるRTLデータパスを合成し，組合せ回路用のテスト生成アルゴリズムを適用することを目的とする．

部分スキャン設計を指向したデータパスのテスト容易化高位合成法として，これまでに多くの手法が提案されている．文献 [14] では一部のレジスタをスキャンレジスタに割り当て，レジスタの可制御性/可観測性を向上させるためのデータパスの合成法が提案されている．文献 [15] では小さい数のスキャンレジスタを用いてセルフループ以外のすべてのフィードバックループを切断するデータパスの合成法が提案されている．文献 [17] ではセルフループ以外のすべてのフィードバックループを切断するスキャンレジスタ数を小さくするためのレジスタのバインディング法が提案されている．また，文献 [16] では部分スキャン設計を想定しないが，フィードバックループ数の小さいデータパスの合成法が提案されている．これらの手法はいずれも順序回路用テスト生成アルゴリズムを用いることを前提にしており，必ずしもすべてのフィードバックループを切断する手法ではない．よって，組合せ回路用テスト生成アルゴリズムのための無閉路部分スキャン設計を指向した合成法にそのまま適用することができない．

本章では，高位合成の部分問題として，スケジュールされた動作記述 (データフローグラフ) に対して，リソース数 (演算器数，レジスタ数) の最小性を満たしながら，生成されるRTLデータパスで無閉路化 (セルフループを含むすべてのフィードバックループを切断) のためのスキャンレジスタ数を最小にする演算器とレジスタのバインディング法を提案する．提案するバインディング法は，1) 動作レベルでセルフループを構成する変数は，RTLデータパスでスキャンレジスタに割り当てなければならない，2) 動作レベルでフィードバックループが発生しても，レジスタを効率よく共有できれば，RTLデータパスでスキャンレジスタ数を減らすことができる，という二つの事実に着目し，1) 演算器・レジスタの共有に

よってセルフループの発生をできるだけ回避する，2) できるだけ多くのフィードバックループが同じレジスタを通るように演算器・レジスタを割り当てるものである．

以下，3.2節で提案するバインディング法の全体の流れを示し，3.3節で演算器とレジスタバインディングの発見的手法について詳細を説明する．3.4節で最小クリーク分割を用いたバインディングのヒューリスティックアルゴリズムについて示し，3.5節で動作記述のベンチマークに対する実験結果より提案手法の有効性を示す．

3.2. 全体の流れ

提案するバインディング法は，リソース数 (演算器数，レジスタ数) を最小にする一般的なアルゴリズムを，無閉路部分スキャン設計のためのスキャンレジスタ数が最小になるように変更したものである．もととなるスキャンレジスタ数最小化を指向しない一般のバインディングアルゴリズムとして，文献 [3, 4] にあるような，両立グラフを用いた手法を採用した．ここではまず，そのアルゴリズムについて説明する．なお，スケジューリングとアロケーションはすでに行われているものとする．入力となるデータフローグラフ (DFG) は以下のように定義される．

定義 3.1 スケジュール済 **DFG(SDFG)** は有向グラフ $G_{sD} = (V_D, E_D, t, s)$ である．ここで，V_D は外部入出力を含む演算を頂点とする集合，$E_D \subset V_D \times V_D$ は変数を辺とする集合，$t : V_D \rightarrow \{op_1, op_2, \ldots, op_n\}$ は演算の型，$s : V_D \rightarrow Z^+ \cup \{0\}$ (非負整数) は演算が実行される制御ステップを表す．

ここでは簡単のため，各演算の実行遅延は 1 制御ステップと仮定する．

バインディングの主要な手続きは，SDFG 中の演算を演算器に割り当てる演算器バインディングと変数をレジスタに割り当てるレジスタバインディングからなる．一般には演算器バインディングとレジスタバインディングに分けて問題を解く．ここでは演算器バインディング，レジスタバインディングの順に行う手法を考える．各バインディングついて，最小個の演算器・レジスタの割り当てを行う

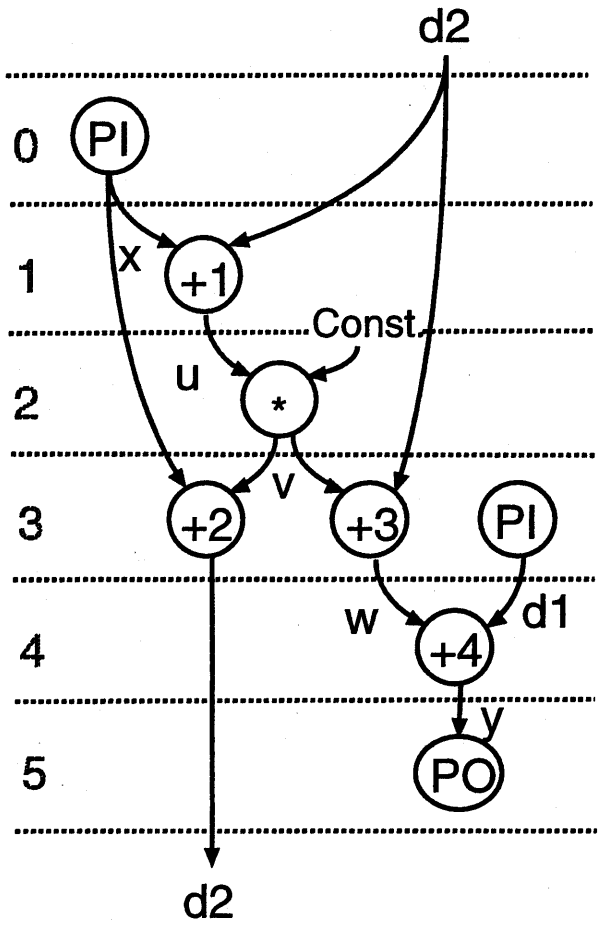

図 3.1 スケジュール済 DFG G_{sD}

ため，両立グラフに対して最小クリーク分割 (クリーク数最小のクリーク分割) を解く．ここで各クリークは共有された演算器またはレジスタに対応している．本章であつかう両立グラフとして，以下で定義する演算/レジスタ両立グラフを用いる．

SDFG 中の二つの演算が同じ制御ステップで実行されず，同じ型の演算器で実現できるとき，これらの演算は**両立可能**であるという．

定義 3.2 SDFG G_{sD} に対する**演算両立グラフ (OCG)** は，無向グラフ $G_O = (V_O, E_O)$ である．ここで，頂点 $v \in V_O$ は SDFG G_{sD} の演算，辺 $(u, v) \in E_O \subset V_O \times V_O$ は頂点 u, v に対応する演算が両立可能であることを表す．

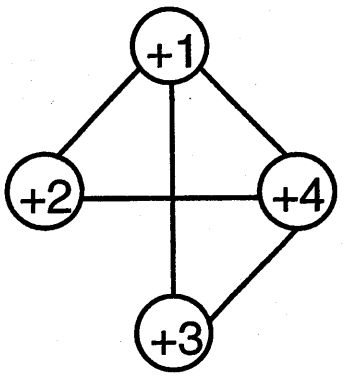

図 3.2 G_{sD}に対する演算両立グラフ G_O

例 3.1 図 3.1 の SDFG の加算に対する演算両立グラフは図 3.2 のようになる．例えば，+1 と +2 はそれぞれステップ 1, 3 でスケジュールされているので，両立可能な辺を持つ．+2 と +3 は同じ制御ステップでスケジュールされているので，その間に辺は存在しない．

SDFG 中の二つの変数のライフタイム (変数が使用されている時間) に重複がないとき，これらの変数は**両立可能**であるという．

定義 3.3 SDFG G_{sD} に対する**レジスタ両立グラフ (RCG)** は，無向グラフ $G_R = (V_R, E_R)$ である．ここで，頂点 $v \in V_R$ は SDFG G_{sD} の変数，辺 $(u, v) \in E_R \subset V_R \times V_R$ は頂点 u, v に対応する変数が両立可能であることを表す．

例 3.2 図 3.1 の SDFG に対するレジスタ両立グラフは図 3.3 のようになる．例えば，変数 u と v はライフタイムが異なるので，両立可能な辺を持つ．一方，変数 $d2$ はすべての制御ステップにわたって利用されているので，それと両立可能な辺は存在しない．

以上で定義した演算/レジスタ両立グラフを用いて，最小クリーク分割により最適なバインディングを求める．最小クリーク分割を求めるとき，演算器数またはレジスタ数に関して等価なバインディングは複数存在することが考えられる．しかし，それらは無閉路化のためのスキャンレジスタ数について必ずしも等価であるとは限らない．複数の最小クリーク分割の解の中からスキャンレジスタ数を

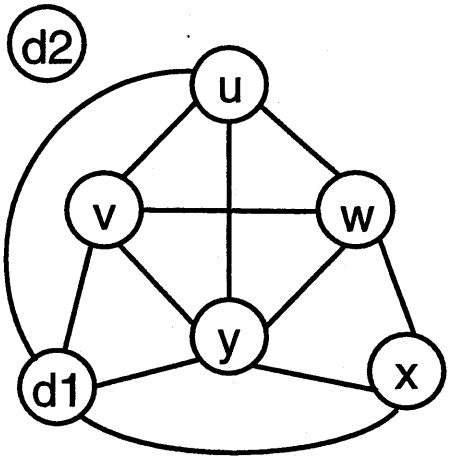

図 3.3 G_{sD}に対するレジスタ両立グラフ G_R

最小にする解を選択するために，スキャンレジスタの必要性をクリークの重みで表す．求めたいバインディングをすべての最小クリーク分割の中で重みが最小になる最小クリーク分割を求める問題として扱うことにする．次の節では演算器とレジスタバインディングのための両立グラフのクリークの重みとその重みを利用した最小クリーク分割について述べる．

3.3. 無閉路部分スキャン設計を指向したバインディング

3.3.1 演算器バインディング

ここでは演算の共有によってできるループを少なくし，それらのループがこの後のレジスタバインディングで互いにスキャンレジスタを共有しやすくすることを考える．

SDFG で両立可能な二つの演算間に経路が存在し，それらの演算を一つの演算器として共有すれば，もとの演算間の経路は共有した演算器を通るループとなる．よって，その演算間の経路上にあるいずれかの変数はループを切断するためのスキャンレジスタに割り当てなければならない．両立可能な演算間の経路の長さ，すなわちその経路上にある変数の数が大きければ，そのうちいずれか一つをスキャンレジスタに割り当てればよいので，スキャンレジスタを選択する自由度

は大きくなる．一般に両立可能な二つの演算間には複数の経路が存在する．簡単のため，ここでは最短経路を複数の経路の代表として扱うことにする．複数ある経路の中で最短経路上の変数は最もスキャンレジスタの共有が行いにくいと考えられる．また，スキャンレジスタに割り当てられる変数のライフタイムが長ければ，スキャンレジスタは共有しにくい．ライフタイムの正確な見積もりはレジスタバインディングで行うとして，ここではライフタイムの代わりに経路を持つ両立可能な二つの演算間の時刻差を単純に評価することにする．

以上のことを考慮して，演算両立グラフの各辺に，以下の重みをつける．

演算両立グラフでつける重み

演算両立グラフの辺 (u,v) の重み：

(1) SDFGで対応する演算 u と v の間に経路が存在しない ($u \rightarrow v, v \rightarrow u$ のいずれの方向にも経路が存在しない) とき：$w_o(u,v) = 0$

(2) SDFGで対応する演算 u と v の間に経路が存在するとき：
u と v の最短経路を p ($u \rightarrow v, v \rightarrow u$ の両方向で経路が存在するとき，それらすべての経路の中で最短)，p の長さと時間 (時刻差) をそれぞれ $l(p)$, $t(p)$ とすると，

$w_o(u,v) = t(p) \times K_{l(p)}$

ここで，$K_{l(p)}$ は $K_{l(p)} > K_{l(p)+1}$ を満たす十分大きな数とする．

上の式を用いると，経路が短いほど大きな重みが与えられる．特に，最短経路長が1のときにできるセルフループは，演算の共有によってできるかぎり作りたくないことを表している．また，重みにかけている最短経路の時間は，同じ長さの最短経路の中だけで差がつくようにしたものである．同じ最短経路長の共有の組合せがあれば，時間の短いものが優先されることを表している．

この重みにおいて，演算間の経路が長い共有については，経路が短い共有と比べて，小さい値の重みが与えられる．したがって実際には，ある程度の長さの経路の短い共有だけで評価に差が十分現われるものと考えられる．実験では長さ5のものまで評価した (3.5節参照)．

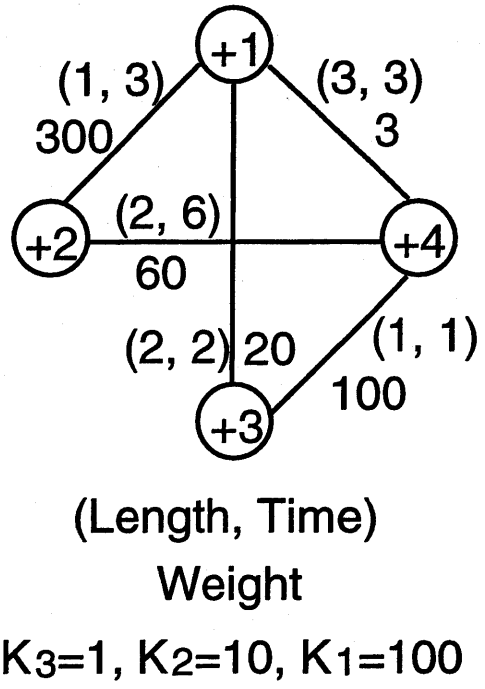

図 3.4 演算両立グラフ G_O の重みつけ

例 3.3 図 3.1 の SDFG の加算についての演算両立グラフ図 3.2 に対して，各辺に対応する演算間の最短経路長とその時間を図 3.4 の括弧の中に示す．ここで，現在のステップ 5 と次のステップ 0 が同じ制御ステップ内で行われるものと仮定する．$K_3 = 1, K_2 = 10, K_1 = 100$ とするとき，各辺の重みは図 3.4 の数値のようになる．例えば，+3 と +4 を一つの演算器として共有すると，共有した演算器はセルフループを構成し，変数 w はスキャンレジスタに割り当てなければならない．同様に +1 と +2 を共有すると $d2$ はスキャンレジスタとなるが，これは w よりも利用されている時間が長い．一方，+1 と +3 を共有すると，セルフループでないループができ，u か v のいずれかがスキャンレジスタに割り当てられる．スキャンのための共有に関しては，(+3, +4) は (+1, +2) よりもよく，(+1, +3) は (+3, +4) よりもよい．図 3.4 の両立グラフの各辺の重みはこの順位を表したものになっている．

クリークの重み

一つの演算器の共有に関するスキャンレジスタの必要性を表す尺度としてクリークの重みを考える．ここでクリークは共有された一つの演算器を表している．演算両立グラフでクリーク (演算器) を構成するとき，重みの大きい辺 (共有) が少ないことが望ましい．よって，クリーク $C_i = (V_i, E_i)$ の重み $W_o(C_i)$ を以下のように定義する．

$$W_o(C_i) = \sum_{e \in E_i} w_o(e)$$

重み和最小クリーク分割問題

演算器バインディングに関するスキャンレジスタの必要性を表す尺度としてクリーク分割の重みを考える．クリーク分割では重みの大きいクリークが少ないことが望ましいことから，クリーク分割の重みをその中に含まれているクリークの重みの総和で表す．演算器数最小のもとで，スキャンレジスタ数を最小にする演算器バインディングとして，以下の問題を考えることができる．

重み和最小クリーク分割問題

入力： 演算両立グラフ $G_O = (V_O, E_O)$

出力： クリークの重みの総和 $\sum_{i=1}^{n} W_o(C_i)$ が最小となるクリーク分割
$\pi = \{C_1, C_2, \ldots, C_n\}$ (クリーク $C_i = (V_i, E_i)$ とすると，$V_O = V_1 \cup V_2 \cup \ldots \cup V_n$ かつ $V_i \cap V_j = \emptyset, \forall i \neq j$)

条件： クリーク数 n が最小

以上のように演算両立グラフに対してクリーク分割を行った結果，演算器の共有を表したグラフとして，同じ演算器に割り当てられる複数の演算に対応するSDFGの頂点を一つの頂点として併合したグラフをつくる．このグラフを**演算共有グラフ**という．

例 3.4 図 3.4 の演算両立グラフに対して，最小となるクリーク数 2 の分割は，
$\{\{+1, +2\}, \{+3, +4\}\}, \{\{+1, +3\}, \{+2, +4\}\}, \{\{+1, +2, +4\}, \{+3\}\},$

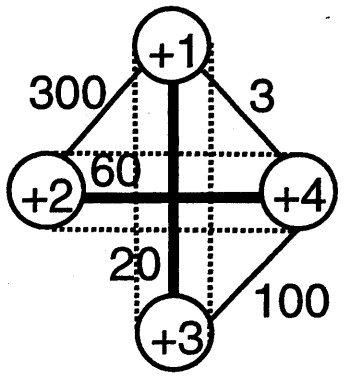

図 3.5 G_Oに対する重み和最小クリーク分割

$\{\{+1,+3,+4\},\{+2\}\}$ の 4 通り存在する．このうち図 3.5 に示す $\{\{+1,+3\},\{+2,+4\}\}$ がクリークの重みの和が $20+60=80$ で最小となる分割である．このときの演算共有グラフは図 3.6 のようになる．結果として，この解はセルフループを含んでいない．一方，ほかの最小クリーク分割の解はすべてセルフループが発生する辺 $(+1,+2)$ または $(+3,+4)$ を含んでいる．

3.3.2 レジスタバインディング

ここでは先に得られた演算共有グラフで変数の共有によってできるループを少なくし，それらのループがスキャンレジスタをできるかぎり共有することを考える．

演算共有グラフで両立可能な二つの変数間に経路が存在し，それらの変数を一つのレジスタとして共有すれば，もとの変数間の経路は共有したレジスタを通るループとなる．よって，その変数自身も含める経路上にあるいずれかの変数はスキャンレジスタに割り当てなければならない．特に，演算共有グラフで隣接している二つの変数を一つのレジスタとして共有すれば，もとの経路は共有したレジスタを通るセルフループとなるので，そのレジスタは必ずスキャンレジスタにしなければならない．演算共有グラフで共有する二つの変数間に隣接以外の経路が

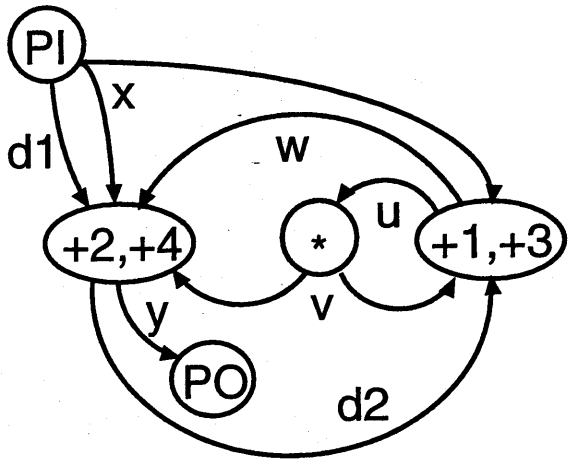

図 3.6 演算共有グラフ G_{oD}

ある場合は，共有したレジスタがスキャンレジスタになるかどうかは，経路中のほかの変数がスキャンレジスタに割り当てられるかどうかに影響する．両立可能な二つの変数を一つのレジスタとして共有したとき，そのレジスタがスキャンレジスタになるかどうかをレジスタ両立グラフの辺の重みで表現する．

また，ほかのどの変数との共有に関係なく，演算共有グラフでセルフループを構成している変数は必ずスキャンレジスタに割り当てなければならない．これをレジスタ両立グラフの頂点の重みで表現する．

レジスタ両立グラフでつける重み

レジスタ両立グラフの各辺に以下の重みをつける．

レジスタ両立グラフの辺 (u,v) の重み：

(1) 演算共有グラフで対応する変数 u と v の間に経路が存在しない ($u \rightarrow v$, $v \rightarrow u$ のいずれの方向にも経路が存在しない) とき：$w_{er}(u,v) = 0$

(2) 演算共有グラフで対応する変数 u と v の間に経路が存在する ($u \rightarrow v, v \rightarrow u$ のいずれかの方向で経路が存在する) とき：

(2-1) 変数 u, v が隣接しているとき：$w_{er}(u,v) = 1$

(2-1) 変数 u, v が隣接していないとき：$w_{er}(u,v) = \omega$ (ただし，$0 < \omega < 1$)

レジスタ両立グラフの各頂点(変数)に以下の重みをつける.

レジスタ両立グラフの頂点 v の重み:

(1) 演算共有グラフで対応する変数 v がセルフループを構成しているとき:
$w_{vr}(v) = 1$

(2) (1) 以外のとき: $w_{vr}(v) = 0$

例 3.5 図 3.3 のレジスタ両立グラフに対して,図 3.6 の演算共有グラフにある演算器バインディングが行われたときの各辺と各頂点に重みをつけたレジスタ両立グラフは図 3.7 のようになる.ここで,$\omega = 0.5$ とする.図 3.6 の演算共有グラフでは,それ自身がセルフループを構成している変数が存在しないので,レジスタ両立グラフの各頂点の重みはすべて 0 となる.各辺の重みについて考えると,例えば,w と x は一つのレジスタとして共有するとセルフループができ,共有したレジスタはスキャンレジスタになる.$d1$ と x は共有してもループができることはなく,スキャンレジスタの必要がない.スキャンのための共有に関しては,$(d1, x)$ は (w, x) よりはるかによい.このように,レジスタ両立グラフの各辺の重みは変数を共有することによるスキャンレジスタの必要性を表している.

クリークの重み

一つのレジスタの共有に関するスキャンレジスタの必要性を表す尺度としてクリークの重みを考える.ここでクリークは共有された一つのレジスタを表している.レジスタ両立グラフでクリークを構成するとき,共有したレジスタ(クリーク)がセルフループができる変数の共有(重み 1 の辺)やセルフループを構成する変数(重み 1 の頂点)を一つでも含んでいれば,そのレジスタはスキャンレジスタでなければならない.クリーク $C_i = (V_i, E_i)$ の重み $W_r(C_i)$ を以下のように定義する.

$$W_r(C_i) = \max\{\max_{e \in E_i} w_{er}(e), \max_{v \in V_i} w_{vr}(v)\}$$

クリークの重みはその中に含まれているすべての変数を一つのレジスタとして共有したとき,そのレジスタがスキャンレジスタになるかどうかを表している.

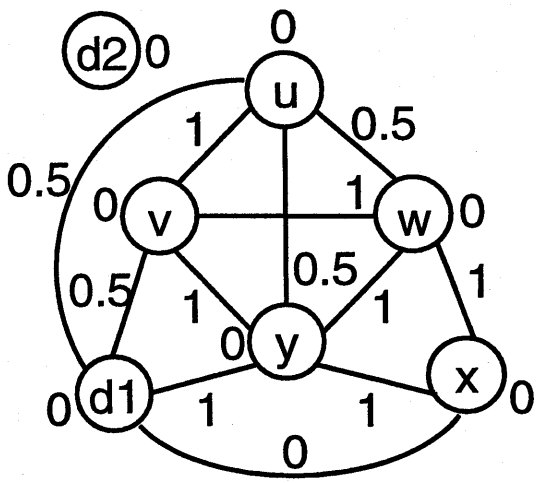

図 3.7 レジスタ両立グラフ G_R の重みつけ

重み和最小クリーク分割問題

レジスタバインディングに関してスキャンレジスタの必要性を表す尺度としてクリーク分割の重みを考える．クリーク分割では重みの大きいクリーク，特にスキャンレジスタに割り当てなければならない重み 1 のクリークが少ないことが望ましい．よって，クリーク分割の重みをその中に含まれているクリークの重みの総和で表す．この重みが小さくなるようにクリーク分割を行えば，必要なスキャンレジスタ数を小さくすることができると考えられる．実際，この重みは結果として RTL データパスで必要なおおよそのスキャンレジスタ数を表している．レジスタ数最小のもとで，スキャンレジスタ数を最小にするレジスタバインディングとして，以下の問題を考えることができる．

重み和最小クリーク分割問題

入力： レジスタ両立グラフ $G_R = (V_R, E_R)$

出力： クリークの重みの総和 $\sum_{i=1}^{n} W_r(C_i)$ が最小となるクリーク分割
$\pi = \{C_1, C_2, \ldots, C_n\}$ (クリーク $C_i = (V_i, E_i)$ とすると，$V_R = V_1 \cup V_2 \cup \ldots \cup V_n$ かつ $V_i \cap V_j = \emptyset, \forall i \neq j$)

条件： クリーク数 n が最小

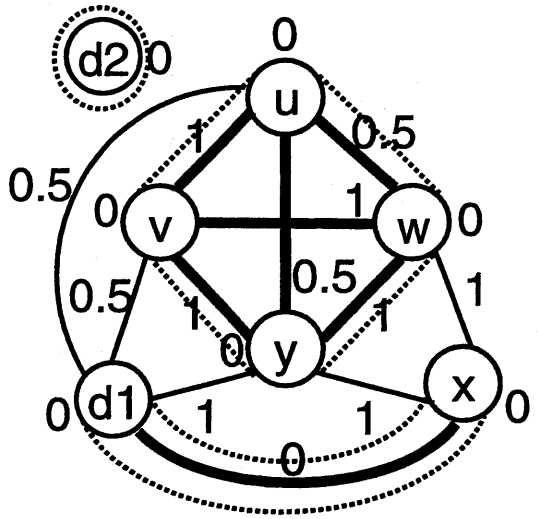

図 3.8 G_Rに対する重み和最小クリーク分割

例 3.6 図 3.7 のレジスタ両立グラフに対して，最小となるクリーク数 3 の分割は，$\{\{d2\}, \{u,v,w\}, \{d1,x,y\}\}$, $\{\{d2\}, \{u,v,w,y\}, \{d1,x\}\}$, $\{\{d2\}, \{u,v,d1\}, \{w,x,y\}\}$, $\{\{d2\}, \{u,v,y,d1\}, \{w,x\}\}$ の 4 通り存在する．このうち図 3.8 に示す $\{\{d2\}, \{u,v,w,y\}, \{d1,x\}\}$ がクリークの重みの和が $0+1+0=1$ で最小となる分割である．このとき，セルフループを構成するクリーク $\{u,v,w,y\}$ によって，少なくとも一つのスキャンレジスタが必要である．これに対し，ほかの最小クリーク分割の解ではセルフループを構成するクリーク (この例では重み 1 の辺を含んでいるクリーク) が二つできるため，少なくとも二つのスキャンレジスタが必要となる．

例 3.7 例 3.4, 3.6 (図 3.5, 3.8) の演算器・レジスタバインディングの結果，合成された RTL データパスは図 3.9 のようになり，無閉路部分スキャン設計のための最小のスキャンレジスタ数は 1 となる．これは図 3.1 の SDFG から合成される RTL データパスの中で，最小のスキャンレジスタ数である．一方，例 3.6 のレジスタバインディングでクリークの重みの和が最小にならないクリーク分割 $\{\{d2\}, \{u,v,w\}, \{d1,x,y\}\}$ では，RTL データパスは図

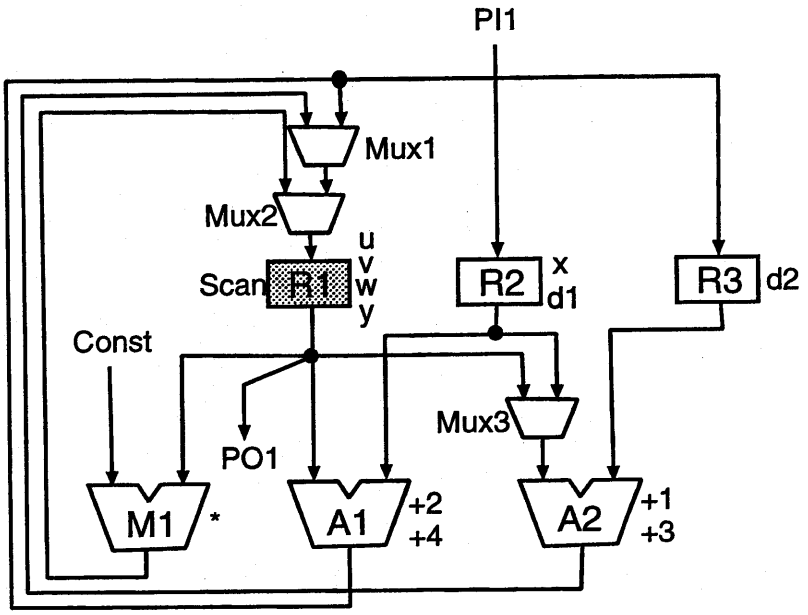

図 3.9 合成された RTL データパス (本手法)

3.10 のようになり，$\{u,v,w\}$, $\{d1,x,y\}$ に対応するレジスタがセルフループを構成することから，少なくとも二つのスキャンレジスタが必要となる．

3.4. ヒューリスティックアルゴリズム

ここでは前節で述べた重み和最小クリーク分割を解くヒューリスティックアルゴリズムを示す．これは重みなしの最小クリーク分割を求めるヒューリスティックアルゴリズム [4] をもとにして，クリーク数最小を求めながらスキャンレジスタ数が最小となるように重みを組み込んだものである．クリークの重みの計算を変えることで，演算器とレジスタのバインディングの両方に同じアルゴリズムを適用することができる．アルゴリズム `weighted_min_clique` を図 3.11 に示す．

演算/レジスタ両立グラフ `Gc` に対して，`weighted_min_clique` はクリーク数最小に関して等価な複数の解の中からスキャンレジスタ数を最小にするクリーク分割 `C_best` を選択する．このアルゴリズムでははじめに各頂点にクリークを割り当てる．両立グラフの辺は共有できるクリークの組を表している．

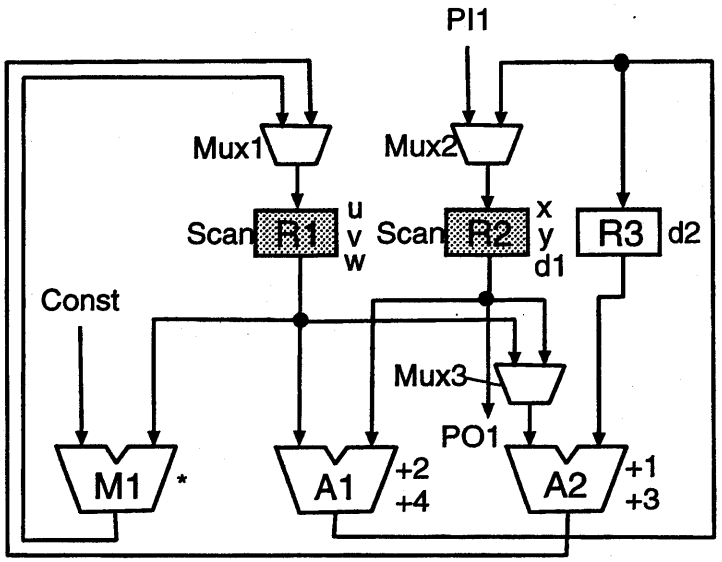

図 3.10 合成された RTL データパス (クリークの重みの和が最小でない場合)

最適なクリーク分割をより広く探索するために，select_start_pair(C) でクリーク分割を複数回繰り返して求め，それらの分割の中からクリーク数が最小でクリークの重みの和が最小になるものを最適解として出力する．select_start_pair(C) では評価関数 H_1 ではじめに同じクリークにできる組を選択し，それらのクリークを共有する．この評価関数は両立グラフの辺の重みが小さいものを優先する．クリーク分割を求める繰り返しは前よりもクリーク数が小さいかまたはクリーク数が同じでクリークの重みの和が小さい分割が求められる限り，続けられる．クリーク分割の解を求める全体の繰り返しの数 Lmax については，実験的に評価する．

クリーク分割を求める各繰り返しについては，評価関数 $H_2(h_c, h_w)$ で同じクリークにできる組を選択し，それらのクリークを共有する．これを同じクリークにできる組がなくなるまで行う．ここでの評価関数は，クリーク数を最小にしながらクリークの重みの和が小さいクリーク分割を求めるのに利用される．具体的には，以下のように考える．

評価尺度 h_c はクリーク数最小の分割を求めるために適用される [4] もので，(α, β) の 2 項組からなる．選択した共有するクリークの組によって，α はほかに共有できる可能性のあるクリーク数，β は共有できなくなるクリーク数を表してい

```
weighted_min_clique(Gc = (Vc, Ec, w_v, w_e))
{
  while (L < Lmax) {
    ++L;
    C = {Vc};
    /* First assign a clique to each vertex */
    (Cs1, Cs2) = select_start_pair(C);
    /* Select a start pair of sharing cliques
       on heuristic function H1 */
    merge(Cs1, Cs2);
    while(Exist a pair of sharing cliques) {
      (Ci, Cj) = select_clique_pair(C);
      /* Select a pair of sharing cliques
         on heuristic function H2 */
      merge(Ci, Cj);
    }
    if ((number(C) < number(C_best)) ||
       ((number(C) == number(C_best)) &&
        (Ws(C) < Ws(C_best)))) {
      C_best = C;
      /* Update solution */
    }
  }
}
```

- Input : compatibility graph `Gc = (Vc, Ec, w_v, w_e)`
  - `Vc` : operation/variable in SDFG
  - `Ec` : sharing possible relation between operations/variables
  - `w_e` : weight of edge
  - `w_v` : weight of vertex
- Output : optimal clique partitioning `C_best`
- `number(C)` : number of cliques
- `Ws(C)` : sum of weights of cliques

図 3.11 重み和最小クリーク分割のヒューリスティックアルゴリズム

る．同じクリークにできる組を選択するとき，α は大きい方が望ましく，β は小さい方が望ましい．評価尺度 h_c は α, β の優先順位に従って評価を行う．

評価尺度 h_w はクリークの重みの和を小さくするためのもので，評価尺度 h_c に関して複数の等価な候補の中から一つを選択するために適用される．共有してもクリークの重みが増えないものを優先する．演算器バインディングでは (選択する両立グラフの辺の重み) − (選択した辺によって共有できなくなる両立グラフの辺の重みの和) を計算し，この値が最も小さいものを選択する．選択した辺によって共有できなくなる両立グラフの辺は，結果としてできるクリーク分割の中に含まれない辺である．この辺の重みを足した値が大きい方が，辺の重みの総和をクリークの重みとするとき，クリークの重みの和を小さくすることができる．レジスタバインディングでは (共有する 2 頂点の重みの和) − (選択する両立グラフの辺の重みと共有する 2 頂点の重みの最大値) を計算し，この値が最も小さいものを選択する．この評価式の第 1 項，第 2 項はそれぞれ共有前と後のクリークの重みの和を表している．この値が 0 以下であれば，クリークの重みが増えることはない．共有前後のクリークの重み和の差が小さいものを優先して選択することにより，辺と頂点の重みの最大値をクリークの重みとするとき，クリークの重みの和を小さくすることを指向している．

3.5. 実験結果

提案手法の有効性を示すために，前節で述べたヒューリスティックアルゴリズムを実装し，いくつかの動作記述のベンチマークに対して演算器とレジスタのバインディングを求める実験を行った．実験に利用したベンチマークは 3rd Lattice Wave Filter (LWF, 図 3.1), Tseng[14], Paulin[14], 4th Jaumann Wave Filter (JWF), 4th IIR Cascade Filter (IIR), 5th Elliptic Wave Filer (EWF) の 6 種類の DFG に対して何通りかのスケジューリングを試みたスケジュール済 DFG(SDFG) である．回路名に付いている '.1' などは同じ DFG でスケジューリングを適当に変化させたものを示している．表 3.1 に使用したベンチマークのレイテンシ (l)，外部入力数 (#PI)，外部出力数 (#PO)，外部入出力以外の演算数 (#Op)，変数の個数 (#Var) を示している．

表 3.1 ベンチマーク特性

Bench.	l	#PI	#PO	#Op	#Var
LWF	5	2	1	5	7
LWF.1					
Tseng	5	3	1	8	11
Tseng.1	6				
Tseng.2					
Paulin	5	4	3	10	11
Paulin.1					
JWF	9	1	1	17	20
JWF.1					
JWF.2					
JWF.3					
IIR	7	1	1	17	22
IIR.1					
IIR.2	8				
IIR.3					
EWF	16	1	1	34	38
EWF.1					
EWF.2					

ここで演算器バインディングにおいて，演算両立グラフの重みは長さ5の最短経路まで評価した．よって，$l(p)$ を最短経路 p の長さとするとき，時間にかける定数は $K_{l(p)} = 32^{5-l(p)}$ とした．レジスタバインディングおいて，重みにつける0と1の間の定数 ω については，0.5とおいて評価した．また，各バインディングでクリーク分割を求める繰り返し数 Lmax については，繰り返しの中で最良の解が得られる回数の適切な値を調べるために，特定の値を設定せずに演算/レジスタ両立グラフの辺数とした．

バインディングの結果得られた RTL 回路に対して，文献 [18] のアルゴリズムを用いて無閉路化に必要な最小個のスキャンレジスタを求めた．その結果を表 3.2 に ST として示す．表 3.2 において，#OU, #Mux, #Reg, #Scan はそれぞれ合成された RTL の演算器数，2入力マルチプレクサ数，レジスタ数，スキャンレジスタ数を表している．CPU は SUN Ultra30 で演算器・レジスタのバインディングに対して Lmax 回の繰り返しでクリーク分割の解を求めるのにかかった CPU 時

表 3.2 ヒューリスティックアルゴリズムによる実験結果

Bench.	M	RTL Characteristics				CPU
		#OU	#Mux	#Reg	#Scan	[s]
LWF	NT	3	6	3	3	<0.1
	ST	3	3	3	1	<0.1
LWF.1	NT	3	5	4	3	<0.1
	ST	3	3	4	1	<0.1
Tseng	NT	7	6	6	4	<0.1
	ST	7	8	5	2	<0.1
Tseng.1	NT	6	8	6	4	<0.1
	ST	6	7	6	3	<0.1
Tseng.2	NT	6	7	6	5	<0.1
	ST	6	10	5	3	<0.1
Paulin	NT	4	17	6	6	<0.1
	ST	4	12	6	4	<0.1
Paulin.1	NT	5	16	7	7	<0.1
	ST	5	13	7	5	<0.1
JWF	NT	3	15	7	7	<0.1
	ST	3	14	7	5	<0.1
JWF.1	NT	3	16	7	7	<0.1
	ST	3	17	7	5	<0.1
JWF.2	NT	3	17	7	7	<0.1
	ST	3	17	7	6	<0.1
JWF.3	NT	4	17	8	8	<0.1
	ST	4	17	8	4	<0.1
IIR	NT	5	21	7	7	<0.1
	ST	5	16	7	4	<0.1
IIR.1	NT	5	23	7	7	1.0
	ST	5	20	7	4	1.0
IIR.2	NT	4	22	7	7	1.0
	ST	4	20	7	4	1.0
IIR.3	NT	4	21	7	7	1.0
	ST	4	13	7	4	1.0
EWF	NT	4	39	11	11	56.0
	ST	4	33	11	7	53.0
EWF.1	NT	4	35	11	11	56.0
	ST	4	37	11	7	53.0
EWF.2	NT	4	35	11	11	56.0
	ST	4	34	11	7	53.0

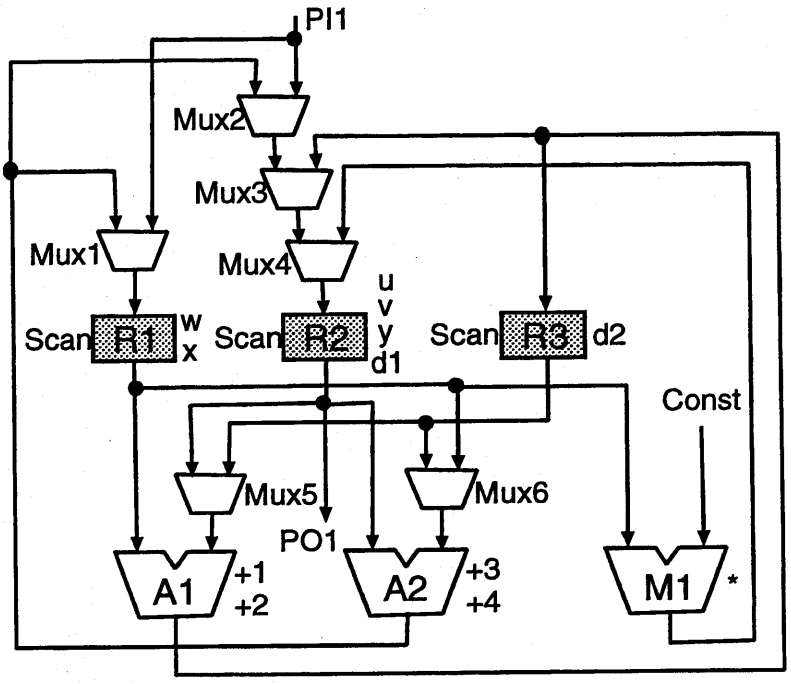

図 3.12 手法 NT による合成結果 (LWF)

間を示している．CPU における '<0.1' は CPU 時間が 0.1 秒未満であったことを示している．

演算/レジスタ両立グラフの重みの効果を調べるために，スキャンレジスタ数最大化を指向する手法 NT も実験した．その結果もあわせて表 3.2 に示してある．NT はクリーク分割の重み (クリークの重みの和) が最大になるようにヒューリスティックアルゴリズムに変更を加えて求めたものである．表 3.2 の結果より，すべてのベンチマークに対して，本手法 ST は手法 NT よりも少ないスキャンレジスタ数が得られた．これらから分かるように，提案した演算器・レジスタバインディングにおける重みは無閉路化のスキャンレジスタ数に相関があると言える．参考のため，図 3.12 に LWF の手法 NT のときの RTL データパスを示す．本手法 ST のときの RTL データパスは図 3.9 に対応している．これらの結果の RTL データパスを見ても分かるように，セルフループ数が減り，複数のループが共通のレジスタを通るようにスキャンレジスタが効率よく共有されて，スキャンレジスタ数が小さくなっている．

表 3.3 重みなしの実験結果

Bench.	RTL Characteristics				CPU
	#OU	#Mux	#Reg	#Scan	[s]
LWF	3	4	3	3	<0.1
LWF.1	3	5	4	3	<0.1
Tseng	7	7	5	2	<0.1
Tseng.1	6	6	6	3	<0.1
Tseng.2	6	8	5	4	<0.1
Paulin	4	12	6	5	<0.1
Paulin.1	5	16	7	5	<0.1
JWF	3	15	7	6	<0.1
JWF.1	3	17	7	7	<0.1
JWF.2	3	13	7	7	<0.1
JWF.3	4	16	8	6	<0.1
IIR	5	17	7	5	<0.1
IIR.1	5	20	7	5	1.0
IIR.2	4	20	7	5	1.0
IIR.3	4	17	7	6	1.0
EWF	4	32	12	10	56.0
EWF.1	4	34	12	10	56.0
EWF.2	4	34	11	10	56.0

クリーク分割の繰り返しの中で最良の解が得られる回数に関しては，ほとんどが 10 以下の少ない回数で，規模の大きい EWT でも両立グラフの辺数 330 の約半分程度の回数でよいことが分かった．

次に，本バインディング法の有効性を調べるために，演算/レジスタ両立グラフで重みをつけずにヒューリスティックアルゴリズムを適用した手法 NW も実験した．その結果を表 3.3 に示す．表 3.2 と 3.3 の結果を比較してみると，ほとんどすべてのベンチマークに対して，NW のスキャンレジスタ数は ST と NT の間の値をとった．ST は NW よりも同じか小さいスキャンレジスタ数が得られていることが分かる．提案した演算器・レジスタバインディング法は有効であると言える．

さらに，ヒューリスティックアルゴリズムの精度を調べるために，表 3.2 の実験とは別に，EWF 以外の例に対して，起こりうるすべてのクリーク分割の中か

表 3.4 全探索による実験結果

Bench.	RTL Characteristics				CPU [s]
	#OU	#Mux	#Reg	#Scan	
Tseng.1	6	7	5	2	<0.1
Tseng.2	6	10	5	2	<0.1
IIR	5	20	7	3	3394.0
IIR.1	5	17	7	3	24059.0
IIR.2	4	10	7	3	16199.0

ら時間をかけて重みが最小の解を求めることを試みた．そのうち文献 [18] のアルゴリズムで求めた最小数のスキャンレジスタが表 3.2 の ST の結果よりさらに小さくなったものを表 3.4 に示す．それ以外のものについては，ヒューリスティックでも演算器数，レジスタ数，およびスキャンレジスタ数は等しく小さい解を得ることができた．この結果から分かるように，Tseng.1 に対する ST (表 3.2) は，スキャンレジスタ数だけでなく，レジスタ数も最小の解を得られていなかった (レジスタ数 6)．しかし，スキャンレジスタに関する重みをつけずに求めた結果 NW (表 3.3) でもレジスタ数は 6 になっていることから，レジスタ数ならびにスキャンレジスタ数の増加は，提案するアルゴリズムのもととした最小クリーク分割のヒューリスティック [4] によるものと思われる．また，IIR, IIR.1, IIR.2 では，ST の解となったレジスタ両立グラフに対するクリーク分割の重みが，表 3.4 に示す解の重みよりわずかに大きい，すなわち，ST のヒューリスティックが最小の重みのクリーク分割を得ていないことが分かった．したがって，重みのわずかな差も表現できるようにヒューリスティックを改良することが課題と言えるが，これらのスキャンレジスタ数は ST においていずれも 4 で，最小の 3 に近い解を選択していることが分かった．Tseng.2 については，$\omega = 0.5$ のもとで，表 3.4 で示したスキャンレジスタ数が 2 となるクリーク分割の重みは，ST で求めたスキャンレジスタ数が 3 となるクリーク分割の重みと等しいことが分かった．すなわち，Tseng.2 については，今回の実験で設定した $\omega = 0.5$ のもとで得られる重み和最小クリーク分割の解は，必ずしもスキャンレジスタ数最小に対応していなかったと言える．この例では ω を 0.5 よりも小さい値にして重みを計算すると，スキャンレジスタ数が 2 のときのみがクリークの重み和が最小となり，ST によってそ

の解が得られることが分かった．ω の適切な値については若干の調整が求められるが，多くの場合，0.5 に設定するだけでよい結果が得られることが分かった．

以上のように，提案する重みづけとヒューリスティックアルゴリズムによって，無閉路化のためのスキャンレジスタ数は最小かもしくはそれに近い値を得ることができると言える．

3.6. むすび

本章では，スケジューリング処理後の動作記述 (データフローグラフ) に対して，テスト容易性を考慮しない従来手法と比較して，演算器数，レジスタ数のリソース数を増やすことなく，無閉路化のためのスキャンレジスタ数の小さいレジスタ転送レベルのデータパスを合成するテスト容易化高位合成手法を提案した．さらに，提案手法を小規模ではあるが動作記述のベンチマークに適用し，その有効性を示した．提案手法はリソース数の最小性を満たしながら，生成される RTL データパスで無閉路部分スキャン設計に必要となるスキャンレジスタ数を最小にする演算器とレジスタのバインディングが得られる．

今後はバインディング手法だけでなく，スキャンレジスタ数最小化のためのスケジューリング手法についても提案する必要がある．さらに，データパスだけでなくコントローラも含めた合成手法についても検討していきたい．

第 4 章

結 論

本論文では，面積オーバヘッドが小さくテスト容易な回路を実現する手法として，順序回路の無閉路構造に基づく部分スキャン設計ならびに高位合成の手法を提案した．

第2章では，順序回路の無閉路構造に基づく部分スキャン設計の一手法として，内部平衡構造に基づく拡張部分スキャン設計法を提案した．ここでは，無閉路構造の中でも組合せ回路のテスト生成アルゴリズムを容易に適用できる順序回路として内部平衡構造を考え，核回路を内部平衡構造にする拡張部分スキャン設計の手法を示し，その性質を明らかにした．拡張部分スキャン設計では，従来の部分スキャン設計と比較して，フリップフロップだけでなく信号線も選択することによって選択の自由度が大きくなり，回路全体の面積オーバーヘッドを小さくすることが期待できる．また，任意に与えられた順序回路から核回路が内部平衡構造となるときの面積オーバーヘッドを小さくするためのフリップフロップおよび信号線を選択する問題について考察した．さらに，核回路を内部平衡構造とする拡張部分スキャン設計された回路では，組合せ回路のテスト生成アルゴリズムで生成されたパターンを変換した系列によってテストすることができることを導いた．最後に，ISCAS'89 ベンチマーク回路に対する実験結果より，提案した拡張部分スキャン設計が小さい面積オーバヘッドで実現できることを示した．

さらに第3章では，テスト容易性を考慮した高位合成 (テスト容易化高位合成) の一手法として，無閉路構造に基づく部分スキャン設計のためのデータパスのテスト容易化高位合成法を提案した．ここでは，スケジューリング処理後の動作記

述 (データフローグラフ) に対して，テスト容易性を考慮しない従来手法と比較して，演算器数，レジスタ数のリソース数を増やすことなく，無閉路化のためのスキャンレジスタ数の小さいレジスタ転送レベルのデータパスを合成するバインディング法を提案した．さらに，提案手法を小規模ではあるが動作記述のベンチマークに適用し，その有効性を示した．提案手法はリソース数の最小性を満たしながら，生成される RTL データパスで無閉路部分スキャン設計に必要となるスキャンレジスタ数を最小にする演算器とレジスタのバインディングが得られる．本合成法の課題としては，バインディング手法だけでなく，スキャンレジスタ数最小化のためのスケジューリング手法についても提案する必要がある．さらに，データパスだけでなくコントローラも含めた合成手法についても検討していくことが挙げられる．

謝辞

本研究の全過程を通じて，常に適切な御指導，御助言を頂いた藤原秀雄教授に心から感謝の意を表します．

本研究の副指導教官として御指導頂きました渡邉勝正教授，福田晃教授に感謝の意を表します．

本研究を進める上で，懇切な御指導，御討論を頂きました増澤利光助教授に深く感謝の意を表します．

本研究の全過程を通じて，方針や問題点等あらゆる面において，懇切丁寧に直接的な御指導を頂きました広島市立大学の井上智生助教授に心から感謝の意を表します．

本研究に際して，色々と御討論下さった井上美智子助手，大竹哲史助手に深く感謝の意を表します．

最後になりましたが，日頃より有益な御討論，並びに，多大な御援助を頂きました藤原研究室の皆様に心から感謝の意を表します．

本研究は一部，(株) 半導体理工学研究センター (STARC) との共同研究，および文部省科学技術研究費補助金・基盤研究 B(2) (課題番号 09480054) の研究助成による．

参考文献

[1] H. Fujiwara, *Logic Testing and Design for Testability*, The MIT Press, 1985.

[2] M. Abramovici, M. A. Breuer and A. D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, 1990.

[3] G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, Inc., 1995.

[4] P. Michiel, U. Lauther, and P. Duzy, *The Synthesis Approach to Digital System Design*, Kluwer Acedemic Publishers, 1992.

[5] K. Cheng and V. D. Agrawal, "A partial scan method for sequential circuits with feedback," *IEEE Trans. on Comput.*, Vol.39, No.4, pp.544-548, April 1990.

[6] D. H. Lee and S. M. Reddy, "On determining scan flip-flops in partial-scan design approach," *Proc. Int. Conf. Computer-Aided Design*, pp.322-325, 1990.

[7] R. Gupta, R. Gupta and M. A. Breuer, "The BALLAST methodology for structured partial scan design," *IEEE Trans. on Comput.*, Vol.39, No.4, pp.538-544, April 1990.

[8] 藤原秀雄, 大竹哲史, 高崎智也, "組合せテスト生成複雑度でテスト生成可能な順序回路構造とその応用," 電子情報通信学会論文誌 DI, Vol.J80-D-I, No.2, pp.155-163, February 1997.

[9] 高崎智也, 井上智生, 藤原秀雄, "内部平衡構造に基づく部分スキャン設計法の考察," 電子情報通信学会論文誌 DI, Vol.J81-D-I, No.3, pp.318-327, March 1998.

[10] T. Inoue, T. Hosokawa, H. Fujiwara, "An optimal time expansion model based on combinational ATPG for RT level circuits," *Proc. IEEE the 7th Asian Test Symposium*, pp.190-197, Dec.1998.

[11] H. J. Wundelich and S. Hellebrand, "The pseudoexhaustive test of sequential circuits," *IEEE Trans. CAD*, Vol.11, No.1, pp.26-33, 1992.

[12] D. Kagaris, S. Tragoudas and D. Bhatia, "Pseudoexhaustive BIST for sequential circuits," *Proc., IEEE Int. Conf. on Computer Design*, pp.523-527, 1993.

[13] J. Steensma, F. Catthoor and H. DeMan, "Partial scan and symbolic test at the register-transfer level," *Journal of Electronic Testing: Theory and Applications*, Vol.7, pp.7-23, 1995.

[14] T. C. Lee, N. K. Jha, and W. H. Wolf, "Behavioral synthesis of highly testable data paths under the non-scan and partial scan environments," *Proc. Design Automation Conf.*, pp.292-297, 1993.

[15] M. Potkonjak, S. Dey, and R. K. Roy, "Behavioral synthesis of area-efficient testable designs using interaction between hardware sharing and partial scan," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol.14, No.9, pp.1141-1154, 1995

[16] A. Mujumdar, R. Jain, and K. Saluja, "Behavioral synthesis of testable designs," *Proc. IEEE Int. Symp. on Fault-Torelant Computing*, pp.436-445, 1994.

[17] V. Fernandez and P. Sanchez, "Partial scan high-level synthesis," *Proc. European Design and Test Conf.*, pp.481-485, 1996.

[18] S. T. Chakradhar, A. Balakrishman, and V. D. Agrawal, "An exact algorithm for selecting partial scan design," *Proc. Design Automation Conf.*, pp.81-86, 1994.