NAIST-IS-DT9861028

博士論文

データパスの強可検査性に基づくテスト容易化設計ならびにテスト容易化高位合成に関する研究

和田 弘樹

2001年3月12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。

和田 弘樹

審査委員：　藤原 秀雄 教授
渡邉 勝正 教授
福田　晃 教授
増澤 利光 教授

データパスの強可検査性に基づくテスト容易化設計ならびにテスト容易化高位合成に関する研究*

和田 弘樹

内容梗概

近年における半導体技術の向上に伴って大規模な論理回路を VLSI として製造することが可能となった．しかし論理回路の大規模化に伴って設計費用および VLSI の信頼性を保証するテストに要する費用の増加が問題となっている．

設計費用の削減手法として計算機援用設計 (CAD) システムを利用した設計の段階的詳細化技法が利用されている．現在，設計者が記述したレジスタ転送レベル (RTL) 回路から論理合成系と呼ばれる CAD システムを用いて目的とする論理回路が設計されている．RTL 回路はデータの処理を行うデータパスとデータパスの動作を制御するコントローラから成る．コントローラは有限状態機械で表現される．データパスは回路要素と回路要素同士を接続する信号線を用いて回路の構造を記述したものである．ここで回路要素とは演算器，レジスタ，マルチプレクサ等の，特定の機能を有する部分回路である．近年では RTL 回路の設計費用を削減する目的で高位合成系が提案されている．高位合成系は設計者によって記述された動作記述を RTL 回路に変換する．動作記述は論理回路の機能を手続きとして記述したものである．

論理回路のテストはテスト生成とテスト実行から成る．故障の有無によって回路の応答が異なるような入力系列 (以下テスト系列) を求めることをテスト生成と呼び，テスト系列を論理回路に印加して得られる応答から故障の有無を判断することをテスト実行と呼ぶ．テスト系列が存在しない故障を冗長故障と呼ぶ．また

* 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-DT9861028, 2001 年 3 月 12 日.

故障検出効率とは回路に存在し得る故障のうち，生成したテスト系列が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である．

記憶素子を含まない組合せ論理回路に対しては，100%の故障検出効率を持つテスト系列を，実用的なテスト生成時間で求めることのできるテスト生成アルゴリズムが提案されている．これに対し，順序論理回路の場合，テスト生成アルゴリズムの探索空間が記憶素子（FF）による内部状態に依存するので，順序回路のテスト生成は一般に困難な問題である．そのため，与えられた回路をテスト生成容易な回路に設計変更するテスト容易化設計法 (DFT) が提案されている．現在広く用いられているテスト容易化設計法には，FF をテストの際に外部から制御および観測可能なスキャン FF に置き換えるスキャン設計法がある．テスト生成の際にスキャン FF を外部入出力と考えることができるので，テスト生成時間を短縮することができるが，過大なハードウエアオーバヘッドの発生等のテスト実行費用の増加が問題とされている．

近年スキャン機構のかわりに回路の通常動作時に利用されるデータの転送経路を用いて回路中の FF の値を制御/観測可能とする非スキャンテスト容易化設計法が提案されている．回路の大部分を占めるデータパスに対して非スキャンテスト容易化設計法を適用した場合，ハードウエアオーバヘッドの増加を抑制しつつテスト容易性が改善可能である．しかし，非スキャンテスト容易化設計法の下でテスト生成費用を削減するためには FF の値の制御及び観測に用いるデータ転送経路が容易に決定できる必要がある．

そこで本論文では RTL データパスのテスト容易性として回路要素の強可検査性を提案する．強可検査である回路要素に対して外部入力から回路要素の入力端子に任意の値を伝達し，伝達された値に対する回路要素の応答を外部出力まで伝達することができる．また個々の回路要素に対して完全故障検出効率を有するテスト系列が容易に生成できるのでデータパス上の全ての回路要素が強可検査であればデータパスに対して完全故障検出効率を有するテスト系列が容易に生成できる．次に任意の RTL データパス上の全ての回路要素を強可検査とするテスト容易化設計を提案する．また演算器の強可検査性が保証された RTL データパスを生成するテスト容易化高位合成法を提案する．これらのテスト容易化設計法およ

びテスト容易化高位合成法で得られる RTL データパスはテスト容易性の向上に伴うハードウアエオーバヘッドの増加が従来手法に比べて小さい.

キーワード

強可検査性, テスト容易化設計, テスト容易化高位合成, レジスタ転送レベルデータパス

Studies on Design and High Level Synthesis for Strong Testability of Data Paths*

Hiroki Wada

Abstract

Along with the improvement in semiconductor technology, the scale of logic circuits is increasing. Also design costs and testing costs of the circuits are increasing.

In order to reduce the design cost, the design of a circuit is incrementally refined by using computer-aided-design(CAD) systems. By using a logic synthesis system, a logic circuit is synthesized from the register-transfer level(RTL) circuit written by designers. An RTL circuit consists of two sub-circuits. One is called a data path which processes data. The other is called a controller which regulates the data path. The data path is represented as a structure consists of modules and signals. Modules are sub-circuits such as an adder, multiplexer or register. The controller is represented as a finite state machine. Recently , high level synthesis systems are proposed in order to reduce the design cost of RTL circuits. By using a high level synthesis system, an RTL circuit is synthesized from the behavioral description written by designers. A behavioral description denotes the function of a circuit in the form of a procedure.

The testing of a logic circuit consists of test generation and test application. Test generation means to generate an input sequence which makes different responses according to the existence of a fault. Such an input sequence is called

* Doctor's Thesis, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT9861028, March 12, 2001.

a "test sequence". Test application means to decide the existence of faults from the responses of a test sequence. If a fault has no test sequence, it is called redundant. Fault efficiency denotes the ratio of the number of faults detected or proved redundant to the total number of target faults under test generation.

For combinational circuits which have no flip-flop(FF), there are several algorithms to achieve 100% fault efficiency with reasonable time. On the other hand, for a sequential circuit, since search space of test generation algorithm depends on the number of internal states represented by FFs of the circuit, sequential test generation is time consuming problem. To ease the complexity of sequential test generation, *design-for-testability* (DFT) techniques are often proposed. The most commonly used DFT techniques are *scan*-based approaches. Scan techniques replaces some FFs in a sequential circuit into *scan* FFs which can be controlled/observed from outside of the circuit. Since these scan FFs can be considered to be primary inputs/outputs, test generation time for a circuit with scan design is greatly reduced. However, scan techniques have disadvantages such as increased test application time, hardware overhead etc.

Recently, non-scan DFT methods are proposed. Values of FFs are justified and observed through data transmission paths used for normal operation. Unlike to scan mechanism, no additional data transmission path is needed. Therefore, the hardware overhead caused by enhancement in testability is suppressed when the DFT is applied to the data path which is the major part of the entire circuit. However, the paths have to be easily selected in terms of the test generation cost.

In this paper, the concept of "strong testability" for a module in a data path is introduced as the condition that the module is easily tested. For the strongly testable module, arbitrary values are justified from primary inputs to the module and arbitrary responses of the module are observed at a primary output. If all modules in a data path are strongly testable, the test sequence with complete fault efficiency can be easily generated. Next, a new DFT method that translates arbitrary data path into strongly testable one is proposed. A new high-level

synthesis for testability method that generates a data path of which all operational modules are strongly testable is also proposed. These proposed methods enhance the testability of resulting data paths with low cost concerned with testing.

Keywords:

strong testability, design-for-testability, high level synthesis for testability , register transfar level data paths

関連発表一覧

- 学術論文誌

  1. 和田弘樹，増澤利光，K.K.Saluja, 藤原秀雄, ”完全故障検出効率を保証するデータパスの非スキャンテスト容易化設計法,” 電子情報通信学会論文誌 (DI), Vol.J82-D-I, No.7, pp.843-851, Jul. 1999 .
  2. 和田弘樹, 増澤利光, 藤原秀雄, ”演算器の強可検査性を保証するテスト容易化高位合成,” 電子情報通信学会論文誌, (採録決定)

- 国際会議 (査読付き)

  1. H. Wada, T. Masuzawa, K. K. Saluja, and H. Fujiwara, ”Design for strong testability of RTL data paths to provide complete fault efficiency,” Proc. of 13th International Conf. on VLSI Design, pp.300-305, Jan. 2000 .
  2. H. Wada, T. Masuzawa, K. K. Saluja, and H. Fujiwara, ”A non-scan DFT method for data paths achieving 100% fault efficiency under hierarchical test environment,” Proc. of European Test Workshop 1999, May, 1999.

- 研究会報告

  1. 和田弘樹, 増澤利光, K. K. Saluja, 藤原秀雄, ”完全故障検出効率を保証する RTL データパスの非スキャンテスト容易化設計法,” 信学技報, (FTS97-44, VLD97-81), Oct. 1997 .
  2. 和田弘樹, 増澤利光, 藤原秀雄, ”強可検査性に基づくテスト容易化高位合成,” 信学技報, (FTS2000-74, VLD2000-109, ICD2000-166), pp.239-244, Nov. 2000 .

目 次

図 目 次

表 目 次

第 1 章
序論

近年における半導体製造技術の向上に伴って，VLSI として製造された大規模論理回路が広く利用されるようになった．また VLSI が広く利用されるようになったため，VLSI に対して高い信頼性が求められるようになった．一般に VLSI の信頼性は故障の有無を調べる**テスト**を行うことで保証される．現在，論理回路の大規模化に伴う論理回路の設計およびテストに要する費用 (以下それぞれ設計費用，テスト費用) の増大が問題になっている．

設計費用を削減する目的で段階的詳細化に基づく回路設計と計算機援用設計 (CAD) システムを用いた設計の詳細化が実用化されている．

現行ではレジスタ転送レベル (RTL) で設計された回路から論理合成系と呼ばれる CAD システムを用いて目的とする論理回路を得る．レジスタ転送レベル回路は入力データを処理するデータパスおよびデータパスの動作を制御するコントローラという 2 つの部分回路から構成される．コントローラは有限状態機械で表現される．データパスは演算器，レジスタ，マルチプレクサ等の，特定の機能を有する部分回路である回路要素を相互に接続した形式で表現される．レジスタ転送レベル回路は機能に基づいて論理回路を抽象化したものである．従って設計者にとって記述が容易であるだけでなく RTL 回路の構成要素数は対応する論理回路の構成要素数よりも小さくなる．

近年では RTL 回路の設計費用を削減する目的で高位合成技術が提案されている．高位合成技術を利用する場合，設計者は論理回路が実現すべき機能を手続き型言語を用いて記述する．設計者が記述した手続き (以下動作記述) は高位合成系と呼ばれる CAD システムを用いて RTL 回路に変換される．動作記述の抽象性と記述性の高さが設計費用の削減に有効であると考えられている．

論理回路のテストはテスト生成とテスト実行から成るのでテスト費用を削減す

るにはテスト生成に要する費用とテスト実行に要する費用を削減する必要がある．ここで，故障の有無に伴って論理回路の応答が異なるような入力系列 (**テスト系列**) を求めることをテスト生成と呼び，テスト系列を論理回路に印加して得られる応答から故障の有無を調べることをテスト実行と呼ぶ．また任意の入力系列に対してその故障の有無に従って回路の応答が変化しないような故障を冗長故障と定義する．一般にテスト系列は自動テストパタン生成系 (ATPG) と呼ばれる CAD システムを用いて生成されるが，計算時間の問題から実用上は故障が冗長であるかどうかの判定を打ち切る場合がある．従って ATPG を用いてテスト生成を行うと回路に発生する可能性のある故障は

1. 冗長でないと判定された (テスト系列が生成できた) 故障
2. 冗長であると判定された故障
3. 冗長であるかどうかの判定が打ち切られた故障

のいずれかに分類される．ここで回路に発生する可能性のある故障の総数を N とし，冗長であるかどうかの判定が打ち切られた故障の総数を N_a とした場合，$\frac{N-N_a}{N}$ を故障検出効率と定義する．特に $\frac{N-N_a}{N} = 1$ の場合を完全故障検出効率と呼ぶ．VLSI の信頼性を確保するためには高い故障検出効率を持つテスト系列を用いたテスト実行を行う必要がある．故障検出効率は

- 対象とする回路
- 故障モデル
- ATPG として実装されているアルゴリズム

等の影響を受けるが，故障モデルとして広く使われている単一縮退故障を用いた場合，次のような事実が知られている．

組合せ回路を対象とした場合: 比較的短時間で完全故障検出効率が得られる (テスト容易性が高い).

順序回路を対象とした場合: テスト生成に長時間を費やしても高い故障検出効率を得ることが困難 (テスト容易性が低い).

ここで単一縮退故障とは論理回路上のある信号線がある論理値に固定されるような故障を指す.

一般の順序回路のテスト容易性を向上する目的で論理回路の設計を変更するテスト容易化設計法 (DFT) が提案されている．一般的な DFT として完全スキャン設計法がある．この手法では論理回路中の全フリップフロップ (以下 FF) をスキャンフリップフロップに置き換えることで, 論理回路上の FF がシフトレジスタとしても動作できるように設計が変更される．FF で構成されるシフトレジスタのスキャン入力とスキャン出力を論理回路の外部入出力端子として追加することで, FF の値を外部から制御/観測可能としている．従って,FF を外部入出力端子で置き換えて得られる組合せ回路 (以下, 核回路) に対してテスト生成を行えば, 完全故障検出効率を持つテスト系列が比較的短いテスト生成時間で得られる．しかし設計変更に伴うハードウエアオーバヘッドやテスト実行時間等のテスト実行に伴う費用が大きく増加するという問題点を有する．

近年スキャン機構のかわりに回路の通常動作時に利用されるデータの転送経路を用いて回路中の FF の値を制御/観測可能とすることでテスト容易性を向上しつつテスト実行に伴う費用の増加を抑制する非スキャンテスト容易化設計法が提案されている．非スキャンテスト容易化設計法では通常動作時に利用されるデータの転送経路を効率良く抽出する目的で RTL データパスを対象としている．

そこで本論文では RTL データパスのテスト容易性として回路要素の強可検査性を提案する．回路要素が強可検査であれば，外部入力から回路要素の入力端子に任意の値を伝達し，伝達された値に対する回路要素の応答を外部出力まで伝達することができる．回路要素単体に対して完全故障検出効率を有するテスト系列を生成することは容易であるので，データパス上の全ての回路要素が強可検査であればデータパスに対して完全故障検出効率を有するテスト系列が容易に生成できる．任意の RTL データパス上の全ての回路要素を強可検査とするテスト容易化設計を第 2 章で提案する．演算器の強可検査性が保証された RTL データパスを生成するテスト容易化高位合成法を第 3 章で提案する．これらのテスト容易化設計法およびテスト容易化高位合成法で得られる RTL データパスはテスト容易性の向上に伴うハードウエオーバヘッドの増加が従来手法に比べて小さい.

第 2 章
テスト容易化設計法

2.1. まえがき

近年における論理回路の大規模化に伴い, 論理回路の故障の有無を調べる**テスト**が重要となり, 故障の有無で論理回路の出力が異なる入力系列 (**テスト系列**) を印加するテストの要求も高まっている. 特に単一縮退故障集合に対して**完全故障検出効率**を持つテスト系列でのテストは, 回路の信頼性を保証する観点からも極めて有用だが, 一般の順序回路に対して故障検出効率が高いテスト系列は生成困難である. そこで, 単一縮退故障集合に対して高い故障検出効率を持つテスト系列が**容易に** (組合せ回路に対するテスト生成と同程度の計算量で) 生成できるように論理回路の設計を変更する**テスト容易化設計** (以下 DFT) が実用化されている.

完全スキャン設計 [1] は現在一般的な DFT であり, ゲートレベル回路中の全フリップフロップ (以下 FF) をスキャンフリップフロップに置き換えることで,FF の値を外部から制御/観測可能としている. 従って,FF を外部入出力端子で置き換えて得られる組合せ回路 (以下, 核回路) に対してテスト生成を行えば, 完全故障検出効率を持つテスト系列が容易に得られるが, 以下の問題点がある.

1: 論理合成後の回路を変更するので, タイミング等の合成時の制約を損なう.

2: ハードウェアオーバヘッドが大きい.

3: テスト系列長が大きい.

4: 実動作速度 (**at-speed**) テストができない.

上記の問題点を解消するために, ゲートレベル回路に変換される前の設計を対象とした DFT が提案されている [7, 9, 10]. ゲートレベルに変換される前の回路

はデータを処理する**データパス**とデータパスの動作を制御する**コントローラ**という 2 つの部分回路で構成され, データパスはレジスタ転送レベル (以下 RTL) 回路として設計される. ここで RTL 回路とはレジスタやマルチプレクサ, 演算器などの回路要素を相互に接続したものを指す. 一般に回路のほとんどはデータパスなので, RTL データパスを対象とした DFT は上記の問題点の解消に大きく貢献することが期待されており, これまでにいくつかの提案が行われている [10, 7, 9]. 本論文でも RTL データパスに対する DFT 手法を提案する. 以下に RTL データパスに対する既知の DFT 手法について述べる.

まず RTL 回路に対して DFT を行うことで, 論理合成後の回路への DFT が不用となるため, 問題点 1 は解消される. 問題点 2,3,4 を解消する目的で, データパスが通常使用するデータ転送経路を通じてレジスタの値の制御/観測を行う DFT として直交スキャン設計法 [10, 11] や Genesis[5, 6, 7, 8, 9] が提案されている. 一般に完全スキャン設計の回路をテストする場合, 実動作時とは別系統のクロック系に実動作時よりも周波数の低いクロックを与える必要がある. つまり, 実動作時に用いられるクロック系に実動作時と同じ周波数のクロックを与えるテスト (**at-speed テスト**) ができない. 直交スキャンや Genesis ではテスト実行時のテストパタンや応答の伝達にデータパスの実動作時のデータ転送経路を利用するので, 実動作時のクロック系に実動作時と同じ周波数のクロックを与えるテストが可能である. よって問題点 4. を解消している.

直交スキャン設計では, データパス上の各レジスタの値を外部入出力端子から一度に制御/観測できるように, レジスタに対するホールド機能と演算器に対するマスク素子を追加する. マスク素子は演算器の入出力間での値の伝達に必要な定数を発生する. 直交スキャンでは,DFT とレジスタの制御/観測のためのデータ転送経路発見にかかる時間計算量が $O(n^2)$ (ただし n はデータパス上の回路要素数) となり, ハードウェアオーバヘッドも完全スキャンより小さい. しかし核回路全体に対して一括してテスト生成を行う必要があり, データパスの大規模化に伴うテスト生成時間の増加は避けられない.

またテスト生成時間を短縮する手法として階層テスト生成法 [3, 4] がある. 階層テスト生成法ではデータパス上の故障 f に対して 2 段階のテスト生成を行なう.

第 1 段階では f が発生する回路要素 M のゲートレベル回路を用いてテストベクタ v を生成する. 第 2 段階では M に対してに RTL 回路を用いたテストプラン生成を行う. **テストプラン**とは外部入力から M の入力へ値を伝達し, また M の出力の値を外部出力まで伝達するためのデータパスへの制御入力の時系列である. 第 2 段階で v のためのテストプラン生成に失敗した場合, 第 1 段階にバックトラックして f に対する新たなテストベクタ v' を生成し, 最終的にテストプラン生成が失敗した場合は f を冗長とみなす.

また階層テスト生成を志向した RTL データパスの DFT として Genesis[5, 6, 7, 8, 9] がある. Genesis では第 1 段階として回路要素毎にゲートレベル回路を用いたテスト生成を行なう. 次に第 2 段階として各回路要素に対して, 外部入力から回路要素の入力へ任意の値を伝達し, また回路要素の任意の値を外部出力まで伝達できるテストプランの生成を試みる. テストプランが存在しない場合には DFT として外部入力から直接値を代入したり, 外部出力で直接値を観測するためのマルチプレクサ (テストマルチプレクサ) と配線を RTL データパス上の適切な回路要素の前後に挿入する. このような手法によって従来手法 [3, 4] で生じる第 2 段階から第 1 段階へのバックトラックがなくなる. しかし, テストプラン探索時のバックトラックは発生するので, データパスの大規模化と共にテストプラン生成時間が急速に増大する. また, テストマルチプレクサと外部入出力への配線は大域的で配線長が増加しやすく, 回路面積も増大しやすい.

そこで本論文では完全故障検出効率を保証するテストが実行可能であり, かつ完全スキャン設計の問題点 1,2,3,4 を解決する RTL データパスに対する DFT 手法とテストプラン生成法を提案する. 提案する手法では RTL データパス上のすべての回路要素に対してテストプランが高速に (バックトラックせずに) 求められるようにマスク素子とレジスタのホールド機能を追加する. また生成されるテストプランは Genesis のテストプランと同様にデータパスの外部入力から回路要素への任意の値の伝達と, 回路要素が出力し得る任意の値のデータパスの外部出力への伝達を保証する. 提案する DFT 手法を適用した RTL データパス上の各回路要素に対してテスト生成の第 1 段階で完全故障検出効率を持つテストベクタを求め, 第 2 段階で求めるテストプランを用いてテストベクタの入力と応答の観測を

行なえばデータパス全体に対して完全故障検出効率を達成できる.

直交スキャン設計では, データパス上のすべてのレジスタの値を同時に制御/観測することを要求する. 一方, 提案手法ではすべてのレジスタを同時に制御/観測する必要はなく, 1 つの回路要素に接続する入力信号線に, 外部入力から任意の値の組を設定し, 出力信号線の値を外部出力で観測できれば十分である. 従って提案方式は, 直交スキャン設計よりも DFT で実現する回路の能力が真に小さいため, 直交スキャン設計よりハードウェアオーバヘッドが小さい. また, マスク素子, ホールド機能の追加は局所的な設計変更であり, Genesis のテストマルチプレクサに比べても提案手法はハードウェアオーバヘッドが小さい. よって提案手法は従来の RTL データパスを対象とした DFT に比べてハードウェアオーバヘッドが小さい.

2.2. 対象とするデータパス

本論文で扱う RTL データパスについて述べる.

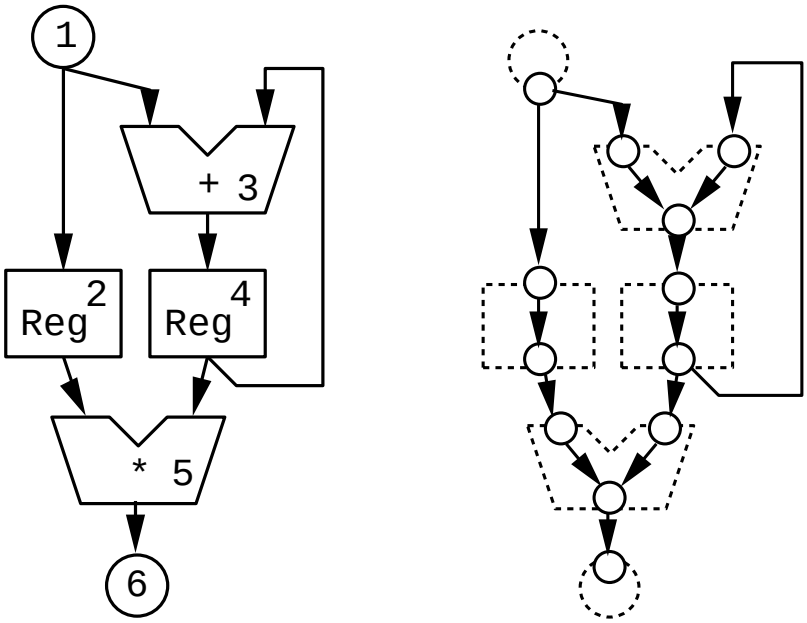
2.2.1 データパス

データパスは信号線と回路要素で構成される. **信号線**は制御信号線, 状態信号線, データ信号線に分類される. **制御信号線**はコントローラからデータパスの回路要素へ制御信号を伝達し, **状態信号線**はデータパスの回路要素からコントローラへ状態信号を伝達する. 本論文では, テスト実行時にすべての制御信号線に対して任意の時点で任意の値が回路外部から設定可能であり, かつ, すべての状態信号線に対して任意の時点で任意の値が回路外部から観測可能であると仮定する[1] .

データ信号線 (以下, 信号線) はデータ出力端子とデータ入力端子を接続する. データ出力端子には複数の信号線が接続できる (ファンアウト可能) が, データ入力端子に接続できる信号線は 1 つとする.

[1] データパスとコントローラから構成される実際のレジスタ転送レベル回路のデータパスに対して本論文で提案するテスト容易化設計法を適用する場合, 制御信号線及び状態信号線を制御/観測するための機構が別途必要となる. このような機能の構成方法及び設計に与える各種影響に関しては文献 [14] 参照

回路要素は外部入力, 外部出力, レジスタ, マルチプレクサ, 演算モジュール, 観測モジュールに分類され, 各種信号線が接続される端子を持つ. データ信号線が接続される端子を**データ端子**, 制御信号線が接続される端子を**制御端子**, 状態信号線が接続される端子を**状態端子**と呼ぶ. データ端子は回路要素の入力であるデータ入力端子 (以下, 入力端子) と, 出力であるデータ出力端子 (以下, 出力端子) に分類される. すべてのデータ端子のビット幅は等しいとする. また, 各入力端子は少なくとも 1 つの外部入力から到達可能であり, 各出力端子からは少なくとも 1 つの外部出力へ到達可能とする.



(a) データパス　　(b) 端子グラフ

図 2.1　データパスと端子グラフ

便宜上, **外部入力**は 1 つの出力端子のみを, **外部出力**は 1 つの入力端子のみを持つ回路要素として扱う. **マルチプレクサ**は 2 つの入力端子と 1 つの出力端子,1 ビットの制御端子を持ち, 制御端子の値によって, いずれかの入力端子の値を出力する.

演算モジュールは 1 つまたは 2 つの入力端子, 1 つの出力端子, 高々1 つの制御端子, 高々1 つの状態端子を持つ. 演算モジュールは, 制御端子と (2 入力演算モ

ジュールの場合は) 他方の入力端子に対して適切な定数を与えることで, 各入力端子と出力端子の間が全単射になると仮定する. つまり, 演算モジュール M が実現する関数族 $\mathcal{F}_M$ に対して, M が 1 入力 (入力端子を x, 出力端子を z とする) ならば, $z = f(x)$ が全単射となる関数 $f \in \mathcal{F}_M$ の存在を仮定する. また M が 2 入力 (入力端子を x, y, 出力端子を z とする) ならば, $z = f(x, a), z = g(b, y)$ がそれぞれ全単射となる関数 $f, g \in \mathcal{F}_M$ と定数 a, b の存在を仮定する. 以降,$f(x, a)$ を $f_{y=a}(x)$ と表し,$g(b, y)$ を $g_{x=b}(y)$ と表す[2] .

観測モジュールは 1 つまたは 2 つの入力端子,1 つの状態端子, 高々1 つの制御端子を持ち, 出力端子を持たない. 観測モジュールは比較器等のモデルである.

データパス上の回路要素 e_1, e_k に対して e_1 を始点,e_k を終点とする経路 p を $p = (e_1, l_1, e_2, l_2, \cdots, l_{k-1}, e_k)$ と表し,e_1–e_k 経路と呼ぶ. ここで l_i は信号線で, 回路要素 e_i の出力端子と回路要素 e_{i+1} の入力端子の 1 つを接続する. 特にすべての e_i が相異なる場合,p を**単純経路**と呼ぶ. 経路 p の始点と終点が同じ回路要素 e であり,e が始点と終点以外に現れず,e 以外の回路要素が p に高々1 回だけ現れる場合, p を**サイクル**と呼ぶ. また, 経路 p に現れるレジスタ数を p の**順序深度**と呼ぶ. 任意の相異なる回路要素 e_1, e_2 に対して, e_1 を始点,e_2 を終点とする任意の相異なる単純経路の組を**再収束経路の組**と呼ぶ.

本論文で提案する DFT 手法が適用できるためにデータパス上の任意の経路が満たすべき条件は次の通り.

P1: 任意の演算/観測モジュール e_1, e_2 を始点および終点とする経路にレジスタが存在

P2: 任意の再収束経路の組に対して, 一方の経路にのみ含まれるレジスタが存在

P3: 任意のサイクルにレジスタが存在する.

[2] 例えば演算器 M が加算器の場合, 関数 f,g はそれぞれ加算演算となる. この場合,f, g に対応する定数の例として 0 がある.

2.2.2 端子グラフ

以下では, データパスを端子グラフ $G = (\mathbf{V}, \mathbf{E})$ として表す. 頂点集合 $\mathbf{V}$ はデータパス上のすべてのデータ端子からなる. また, 有向辺集合 $\mathbf{E} = \mathbf{E}_1 \cup \mathbf{E}_2$ は信号線に対応する有向辺集合 $\mathbf{E}_1$ と, 回路要素でのデータ端子間の入出力関係を表す有向辺集合 $\mathbf{E}_2$ からなる. つまり, 入力端子 v_1 と出力端子 v_2 が同じ回路要素に属する場合のみ, 有向辺 (v_1, v_2) が $\mathbf{E}_2$ の要素となる.

図 2.1 にデータパスと対応する端子グラフの例を示す. 以下では特に混乱のない限り,$\mathbf{V}$ の要素を入出力端子として扱い, $\mathbf{E}_1$ の要素を信号線として扱う.

2.3. テスト容易化設計アルゴリズム

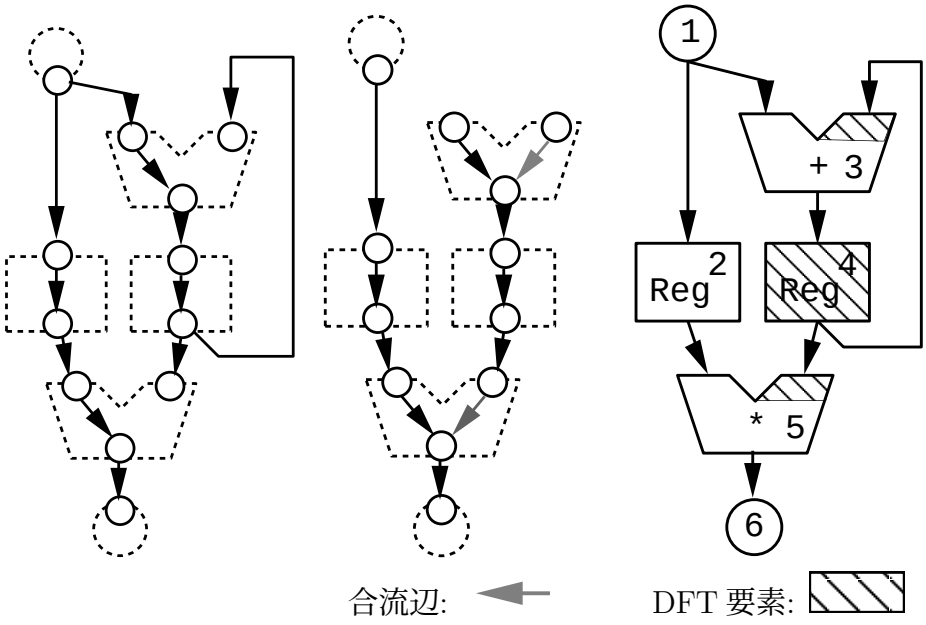
2.3.1 基本方針

本論文ではデータパスに対して階層テスト生成を行なった場合に完全故障検出効率が得られるための十分条件としてデータパスの**強可検査性**を考える. すべての回路要素に対して, 1) 外部入力から任意の値を入力端子に伝達可能 (強可制御性) かつ, 2) 出力端子の取り得る任意の値が外部出力まで伝達可能 (強可観測性) ならば, データパスが強可検査であるという. データパスが強可検査ならば, 全回路要素に対してテストプランが存在することから, 回路要素毎に完全故障検出効率を持つテストベクタ集合を用いることで完全故障検出効率を実現する階層テストが可能となる.

提案するテスト容易化設計法はデータパスを強可検査とする. まず強可制御性を確保するために, 入力端子 a 毎に外部入力から a へ任意の値を伝達する経路 (**制御経路**) を決定する. 次に強可観測性を確保するために, 出力端子 b 毎に b から外部出力まで任意の値を伝達する経路 (**観測経路**) を決定する. そして, これらの経路上で値が伝達できるように **DFT 要素**を追加する. DFT 要素は 2 入力演算モジュールでの全単射の実現に必要な定数を発生する**マスク素子**と経路間のタイミングに関する問題を解決するレジスタの**ホールド機能**からなる.

2.3.2 制御林の生成

各入力端子に対し, 制御経路を求める. 一般に, 外部入力からある入力端子への経路は複数存在するが, テスト実行時間を短縮するために, 順序深度が最小の経路を制御経路とする. また, 追加される DFT 要素を削減する目的で, 制御経路の集合が外部入力を根とする林を構成するように制御経路を選ぶ. 制御林は, データパス上で外部入力を始点として順序深度に基づいた最短経路林を構成することで $O(n)$ 時間で求められる. 図 2.2(a) に制御林の例を示す.



(a) 制御林　(b) 観測林　(c) データパス (DFT 後)

図 2.2 制御/観測林と DFT 後のデータパス

2.3.3 観測林の生成

各出力端子に対し, 観測経路を求める. 一般に, 出力端子から外部出力への経路は複数存在するが, 観測経路と制御経路の共有で DFT 要素が削減されるので, 観測経路と制御経路の共有部分の極大化を目的として, 制御林に含まれない観測経路上の辺が最小となるように選ぶ. (制御林に含まれない観測林上の辺を**合流辺**と

呼ぶ). 観測林は, データパス上で外部出力を始点として制御林に含まれない辺の数に基づいた最短経路林を構成することで $O(n)$ 時間で求められる. 図 2.2(b) に観測林の例を示す.

2.3.4 DFT 要素の追加

任意の制御/観測経路上を任意の値が伝達できるように, 以下のように DFT 要素を追加する.

マスク素子の挿入: 回路要素に関する条件から, すべての回路要素で入出力間に全単射が実現できる. 従って, 制御林上の任意の経路 p において,p 上のすべての辺 $l \in \mathbf{E}_2$ を全単射とする制御入力を回路要素に与えれば p の始点–終点間で任意の値が伝達できる. ただし,2 入力演算モジュール M(入力を x, y, 出力を z とする) に含まれる p 上の辺 e[3] を全単射とする場合,M が全単射 $z = f_{x=a}(y)$ を実現するように,M の制御入力に対して M の関数 $f \in \mathcal{F}_M$ を実現するための定数を与え, x に対して定数 a を与える必要がある. そこで定数 a を発生することが可能なマスク素子を x の直前に追加する. **マスク素子**は 1 ビットの制御入力を持つ 1 入力 1 出力の回路要素で, 制御入力に従って定数または入力値を出力する. マスク素子で定数 a を発生し,M に制御信号として f を与えれば $y - z$ 間が全単射になる. 以下ではマスク素子と M を併せて 1 つの演算モジュール $M^\star$ と見なし, マスク素子と M の制御信号線を併せて $M^\star$ の制御信号線と見なす ($M^\star$ の入力端子を x', y', 出力端子を z' とすると,y' は y と,x' はマスク素子の入力と,z' は z と同一である). また以下では $M^\star$ の関数で, 制御林に含まれる辺 (この場合は $y' - z'$ 間) を全単射とするものを $THRU$ と呼び, 制御林に含まれない辺 (この場合は $x' - z'$ 間) を全単射とする関数を $THRU^\star$ と呼ぶ.

観測林上の合流辺を含まない部分経路 q は制御林に含まれるので, q 上でも値の伝達が可能である. また任意の合流辺 d(d を含む回路要素の入力端子を

[3] 2 入力モジュール上の 2 本の辺のうち 1 本のみが制御林に含まれるので, 一般性を失うことなく $e = (y, z)$ と仮定する.

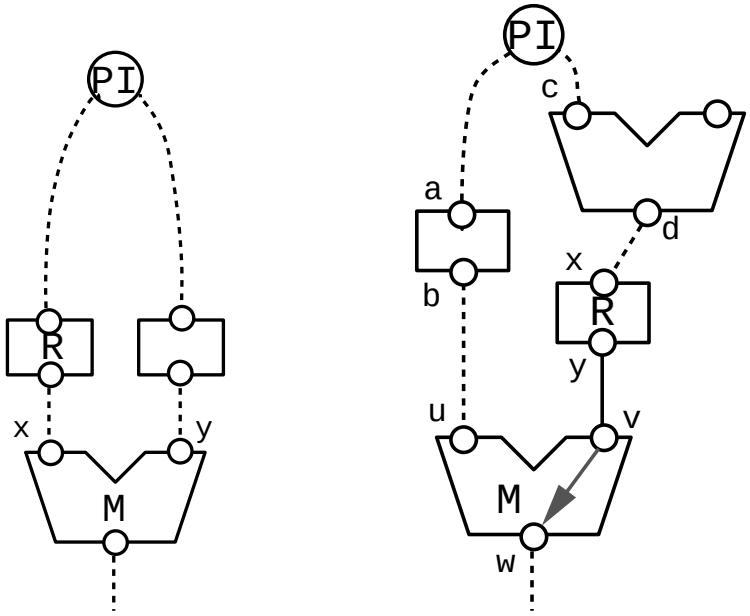
x, y とし,d の始点を x とする) に対し,d を全単射とするためには,y へある定数を入力しなければならないが, この定数は y への制御経路 (以下では,d の**補助経路**と呼ぶ) から設定できる. よって制御林上の経路において任意の値の伝達が可能であれば, 任意の観測経路においても任意の値が伝達される.

各経路上で値を伝達する方法を以下にまとめる. 制御経路上ではレジスタに値をロードし, マルチプレクサには経路上の入力端子の値を出力する制御信号を与える. また演算モジュールには関数 $THRU$ を実現するために必要な定数を制御入力に与える. 観測経路上ではレジスタに値ををロードし, マルチプレクサには経路上の入力端子の値を出力する制御信号を与える. また合流辺でない辺を経路上に持つ演算モジュールにはその関数 $THRU$ を実現するために必要な定数を制御入力に与える. 一方で合流辺を経路上に持つ演算モジュールには関数 $THRU^{\star}$ を実現するために必要な定数を制御入力と他方の (経路上にない) データ入力に与える. このときデータ入力に与える定数は合流辺の補助経路を用いて外部入力から供給する.

ホールド機能の追加: テストベクタを印加する際やテストベクタに対する回路要素の応答を伝達する際に**タイミング衝突**が発生する場合がある. ここでタイミング衝突とは, ある時点で同一の頂点に異なる値を与えようとすることを指し, テストに必要な経路の集合が, 順序深度の等しい経路からなる再収束経路の組を含む場合に再収束経路の始点で発生する. 経路に関する条件 (P2) から再収束経路の組にはどちらか一方にしか含まれないレジスタが存在する. このレジスタにホールド機能を与えることで再収束経路の組を構成する経路間でテスト実行時の順序深度が実質的に異なるようにする. タイミング衝突が発生するのは次のような場合である.

場合 1: 2 入力の回路要素 M(入力を x, y とする) のテスト時に, x の制御経路と y の制御経路が同一の始点と順序深度を持つ場合 (図 2.3(a)).

場合 2: 2 入力の回路要素 M(入力を u, v, 出力を w とし (v, w) を合流辺とする) 上の合流辺 (v, w) を含む観測経路 s を用いてテストベクタに対する応答を観測する場合を考える. 辺 (v, w) の補助経路同じ始点および



(a) (b)

図 2.3 ホールド機能の追加

順序深度を有し, かつ v を終点とする経路が s と制御経路を接続して得られる経路上もしくは s と s 上の合流辺に対する補助経路を接続して得られる経路上に含まれる場合 (図 2.3(b)). 図 2.3(b) において経路 $PI-a-b-u$ が経路 s に相当する. また s と制御経路を接続して得られる経路上もしくは s と s 上の合流辺に対する補助経路を接続して得られる経路上に存在する経路で辺 (v,w) の補助経路同じ始点および順序深度を有し, かつ v を終点とするものは経路 $PI-c-d-x-y-v$ に対応する.

回路要素 M の入力端子 x に対して $depth(x)$ は x への制御経路の順序深度を, $ctrlPI(x)$ は x への制御経路の始点となる外部入力を表すこととする. また合流辺 (v,w) のうち $depth(v)$ と $depth(w)$ が異なるものを観測林から除いてできた林を T' とする. 以下のいずれかの条件を満たすレジスタにホールド機能を与える.

- 任意の 2 入力の回路要素 M(入力端子を x, y, 出力端子を z とし, (x, z) が制御林に含まれないものとする) において, $depth(x) = depth(y)$ かつ $ctrlPI(x) = ctrlPI(y)$ である場合, 制御林上で x に最も近い先祖のレジスタ R にホールド機能を与える (図 2.3(a)).
- 任意のレジスタ R(入力を x, 出力を y とする) に対して以下の条件を満たす回路要素 M(入力を u, v, 出力を w とし,(v, w) を合流辺とする) が存在する場合, R にホールド機能を与える (図 2.3(b)).
  - $depth(u) = depth(v)$
  - T' 上で順序深度が 0 の y–v 経路が存在
  - $ctrlPI(u) = ctrlPI(x)$

このテスト容易化設計により, 図 2.1(a) のデータパスから図 2.2(c) のデータパスが得られる. 図 2.2(c) の斜線部にホールド機能またはマスク素子が挿入される.

2.4. テストプラン生成アルゴリズム

本節では前節のテスト容易化設計で得られたデータパス上の任意の回路要素 M に対し, そのテストプランを生成するアルゴリズムを示す. ただし, アルゴリズムの入力として, 制御林, 観測林も与える.

M が 2 入力の回路要素で,M を通るサイクルが存在する場合, M の入力端子に対する制御経路 (cp とする) の中間に M が現れることがある. M が故障している場合,cp を用いてテストベクタが正しく伝達できず, M の故障が検出されない可能性がある. そこで, このような cp が存在する場合, テストベクタ毎に**副テスト**を行ってから**主テスト**を行う. ここで副テストとは外部入力から cp を通して外部出力までテストベクタを伝達することで cp 上をテストベクタが正しく伝達されるかを確認するテストであり, 主テストとは cp を用いた M のテストを指す. 副テストのためのテストプランは主テストのためのテストプランと同様に生成できるので, 以下では主テストのためのテストプラン生成についてのみ述べる.

1. **主経路集合の生成:** まず, M の入力端子毎に制御林から制御経路を求め,(も

しあれば)M の出力端子に対して観測林から観測経路を求める. さらに観測経路上の合流辺毎に制御林から補助経路を求める. これらの経路集合を主経路集合と呼ぶ.

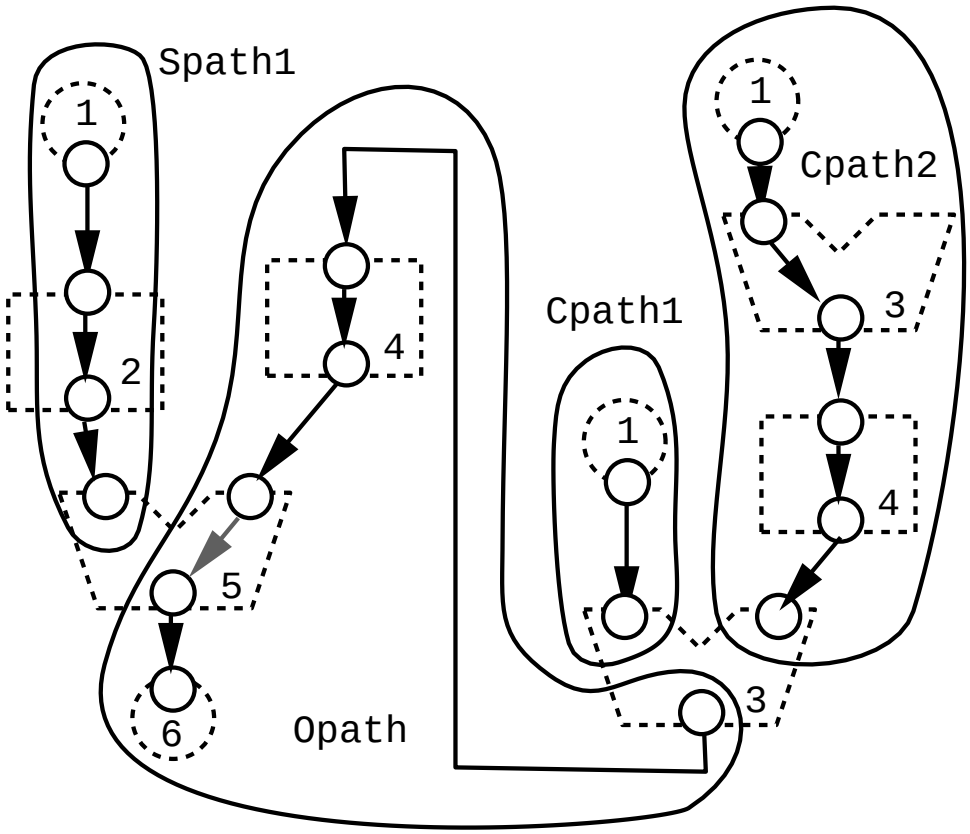


図 2.4 加算器 (3) の主経路集合

図 2.1(a) の加算器に対する主経路集合を図 2.4 に示す. Cpath1,Cpath2 が制御経路,Opath が観測経路,Spath1 が補助経路を表す.

2. **経路集合のタイミング調整:** 3.4 節で示したように, 回路要素 M にテストベクタを外部入力から伝達するとき, あるいは, その応答を外部出力まで伝達するときに, タイミング衝突が発生する可能性がある (例えば図 2.4 において $Cpath1$ と $Spath1$ が同じタイミングで同じ外部入力を利用しようとして

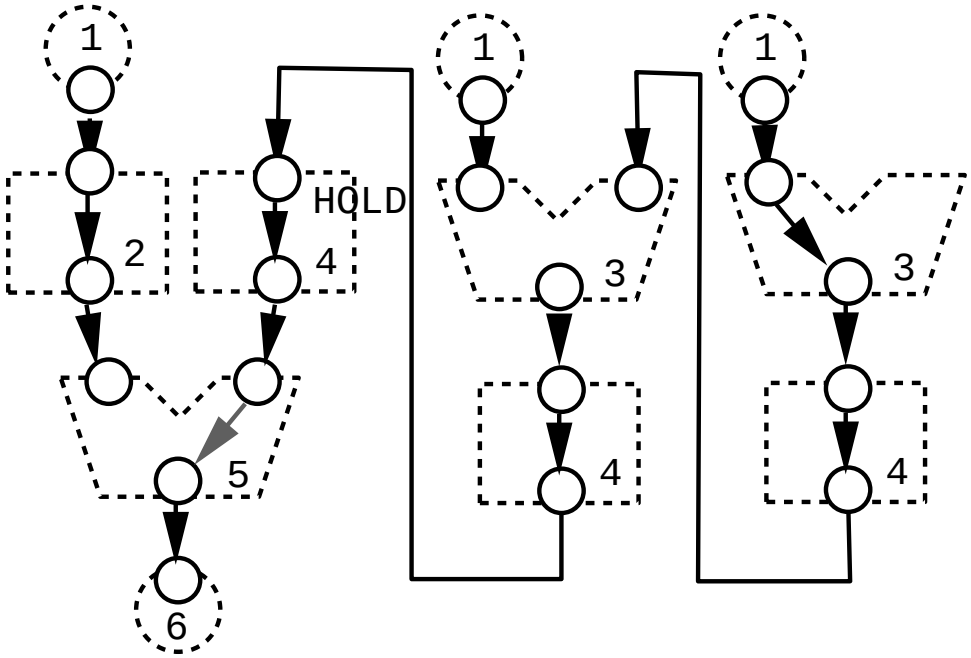


図 2.5 加算器 (3) の主経路集合 (タイミング調整済み)

いる). このようなタイミング衝突を解消する手続きを以下に示す.

1. **制御経路の調整:** M が 2 入力の回路要素 (入力を x, y, 出力を z とし, 一般性を失うことなく (x, z) が制御林に含まれないものと仮定する) である場合, x の制御経路を cp_x,y の制御経路を cp_y とする. cp_x と cp_y の順序深度と始点が共に等しい場合タイミング衝突が生じる. この場合,cp_x 上の終点にもっとも近いレジスタ r にホールド機能が追加されているので,r を 1 回ホールドすることでタイミング衝突を解消する. 以降, 追加されたホールドの回数を経路のレジスタ数に加えたものをその経路の順序深度とする.

2. **補助経路の調整:** 制御経路の調整が終了した後に行なう. 観測経路上の合流辺を始点に近いものから順に $j_1, \cdots, j_k$ とし, 各 j_i の補助経路を sp_i とする. また,j_0 を制御経路と観測経路を結ぶ M 上の辺で制御林に含まれるものとする. $sp_1, \cdots, sp_k$ を順に次の方法で調整する. これまでに

調整が終了した制御/補助経路 l 毎に, 観測経路の始点から j_i の始点までの観測経路上の部分経路を l の終点に連結して得られる経路の集合を $\mathbf{L}$ とし, $\mathbf{L}$ の要素のうち sp_i と始点が等しい経路の集合を $\mathbf{L}^\star$ とする. $\mathbf{L}^\star$ の中に sp_i と順序深度が等しい経路が存在する場合, j_i の始点から観測経路を逆に辿って最初に現れるレジスタ r_i を探す. 1)r_i は j_{i-1}–j_i 間に存在すること, 2) テスト容易化設計でホールド機能が r_i に追加されること, 3) 制御経路は順序深度が最小であることより,r_i を 1 回ホールドする (これによって $\mathbf{L}$ に含まれるすべての経路の順序深度が 1 増える) ことで sp_i とタイミング調整済である経路間のタイミング衝突が解消される.

ただし sp_1 の調整時に観測経路の先頭まで探索してもレジスタが存在しないことがある. この場合, 経路集合に含まれる制御経路が 1 本であれば, 制御経路を終点から逆方向に探索して最初に現れるレジスタを r とする. 経路集合に含まれる制御経路が 2 本であれば, 制御林に含まれない M 上の辺の始点を終点に持つ制御経路 cl が存在するので, そちらを終点から逆方向に探索し, 最初に現れるレジスタを r とする. この場合,r のホールドで順序深度が増える経路は cl のみであるので, cl の順序深度が $\mathbf{L}^\star$ に含まれるどの経路とも異なるような最小の回数だけホールドする.

タイミング調整手続きは経路集合上の辺集合 $\mathbf{J} = (j_0, \cdots, j_i)$ の各要素毎に, 対応する補助経路と既にタイミング調整済みの経路との間でタイミング衝突が起きるかどうかを判定し, もし衝突が起こるのであれば, それを解消する. $\mathbf{J}$ の要素数は高々 n であり, 各要素について衝突判定の時間計算量は $O(n)$ である. 従ってタイミング調整手続きの時間計算量は $O(n^2)$ となる. 図 2.5 に図 2.4 をタイミング調整したものを示す. Spath1 と Cpath1 によって生じるタイミング衝突が Opath 上のレジスタ 4 をホールドすることによって解消されている.

3. **テストプランの生成:** タイミング調整済みの主経路集合と (もしあれば) 副経路集合からテストプランを作成する. 経路集合において, 観測経路の終点

(M が観測モジュールの場合は M) から経路を逆に辿った際のレジスタ数を各回路要素に与える. 与えられた順序深度が大きいものから順に 3.4 節で述べた制御信号を与えることでテストプランが生成できる. 表 2.1 に図 2.5 に対応するテストプランを示す.

時刻	1		
回路要素 制御入力	1 (ベクタ入力)	3 $THRU$	4 ロード
時刻	2		
回路要素 制御入力	1 (ベクタ入力)	3 (テストベクタ)	4 ロード
時刻	3		
回路要素 制御入力	1 (ベクタ入力)	2 ロード	4 ホールド
時刻	4		
回路要素 制御入力	5 $THRU^{\star}$	6 (ベクタ観測)	

表 2.1 テストプラン

2.5. 実験結果

完全故障検出効率を保証するテスト容易化設計法として, 提案方式と完全スキャンを, 実験によって比較した. ４種類のデータパス「GCD」,「4thIIR」,「Paulin」および「RISC プロセッサ」に対して, テスト生成時間, テスト容易化設計に基づくオーバヘッド, テスト系列長の 3 項目を評価した. 表 2.2 に各データパスの特性を示す.

テスト生成時間: 完全スキャン (以下 FS[4]) では初めにデータパス上の全レジスタを外部入出力で置き換えた組合せ回路から論理合成ツールを用いてゲートレベ

[4] Full Scan

	GCD	4thIIR	Paulin	RISC プロセッサ
入力数	2	1	2	1
出力数	1	1	2	3
レジスタ数	3	12	7	40
MUX 数	3	3	8	84
演算モジュール数	2	5	7	19
観測モジュール数	3	0	0	5

表 2.2 データパスの特性

ル回路を得た. 次にゲートレベル回路に対してテスト生成ツールを用いて組合せテスト生成を行ないテストベクタを得た. テスト生成に要した時間を完全スキャンのテスト生成時間 (T_{FS}) とした.

提案方式 (以下 ST[5]) では, 初めにレジスタ転送レベル回路に対してテスト容易化設計とテストプランの生成を行ない, 要した時間を T_1 とした. ここでマルチプレクサ (以下 MUX) と演算/観測モジュールをテストプラン生成の対象とした. 次に論理合成ツールを用いて MUX と演算モジュール (テスト容易化設計済みのもの) , 観測モジュール毎にゲートレベル回路を得た後, 個々のゲートレベル回路に対してテスト生成ツールを用いて組合せテスト生成を行ないテストベクタを得た. 各ゲートレベル回路のテスト生成に要した時間を T_2 とし, 提案方式のテスト生成時間 T_{ST} を $T_1 + \sum T_2$ とした.

なお, 論理合成ツールとして Autologic II(Mentor Graphics), テスト生成ツールとして TestGen(Sunrise) を用い, SUN SPARCstation 20(SuperSPARC 75MHz) 相当の計算機上で実行した (以下の実験も同じ). 結果を表 2.3 に示す.

大規模な回路において提案方式のテスト生成時間は完全スキャンよりも小さい. これはテスト生成を回路全体に対してではなく回路要素毎に行なうことで, テスト生成ツールが取り扱う回路の規模が小さくなることが原因であると思われる. よってこの傾向はデータパスの大規模化と共により顕著になると思われる.

[5] Strongly Testable

	GCD		4thIIR		Paulin		RICS プロセッサ	
	$T_1 = 0.2$		$T_1 = 0.2$		$T_1 = 0.2$		$T_1 = 1.4$	
bit	T_{ST}	T_{FS}	T_{ST}	T_{FS}	T_{ST}	T_{FS}	T_{ST}	T_{FS}
8	1.1	0.5	0.8	0.7	1.2	1.4	–	–
16	1.6	1.7	1.3	2.2	2.8	4.5	–	–
32	5.8	7.5	14.2	33.7	11.3	18.2	58.6 [6]	51740.9 [7]

表 2.3 テスト生成時間 (単位:秒)

テスト系列長:テスト生成で得られるテストベクタとテストプランから以下の方法でテスト系列長を求めた. ここでテスト系列長とは全テストベクタの入力と, その応答の観測に必要なクロック数である. 結果を表 2.4 に示す.

- 提案方式 (ST)
  (回路要素のテストベクタ数 × 回路要素のテストプラン長) の全テスト対象回路要素に対する和

- 完全スキャン (FS)
  (テストベクタ数) × (スキャンパス長 + 1) + (スキャンパス長)
  ただし,(スキャンパス長) = (フリップフロップ数)

完全スキャン方式と提案方式のテスト系列長の差はデータパスのビット幅の増加に伴って急速に広がっている. よって, 提案方式が大規模なデータパスでのテスト実行時間を大幅に削減することが分かる.

ハードウェアオーバヘッド: テスト容易化設計前 (org.), 完全スキャン設計 (FS) 後, 提案方式によるテスト容易化設計 (ST) 後の回路を論理合成し, ゲート数を比較した. ここで, レジスタにホールド機能を与えるために 1bit あたり 3 ゲート, ス

[6] 故障検出効率 99.99%を得るのに有した時間.(テスト生成に十分な時間を掛けても完全故障検出効率が得られなかったため)

[7] 故障検出効率 99.97%を得るのに有した時間.(テスト生成に十分な時間を掛けても完全故障検出効率が得られなかったため)

	GCD		4thIIR		Paulin		RISC プロセッサ	
bit	ST	FS	ST	FS	ST	FS	ST	FS
8	254	1299	852	2715	1137	2792	–	–
16	470	5732	1392	8105	1503	8587	–	–
32	760	22794	2600	71224	2512	24524	8505	1006154

表 2.4 テスト系列長 (単位:クロック)

キャン機能を与えるために 3 ゲート, ホールド機能とスキャン機能を同時に与えるために 4 ゲートを要した. 結果を表 2.5 に示す. ただし, 表中の%OH の項は次の通り.

$$\frac{\text{DFT 後のゲート数} - \text{DFT 前のゲート数}}{\text{DFT 前のゲート数}} \times 100(\%)$$

提案方式は完全スキャン方式に比べて小さいハードウェアオーバヘッドで完全故障検出効率を保証している. 4thIIR に対する提案方式でのオーバヘッドが他の回路に対する提案手法のそれに比べて大きい. これは 4thIIR に含まれる定数乗算器 (1 入力 1 出力の演算モジュール) の入出力間においてマスク素子と制御入力の制御だけでは全単射が実現できないために, 乗ずる定数として設計値の他に定数 1 を選択できるようにして入出力間で全単射が実現できるように機能を拡張したためである.

提案方式では演算モジュールの入出力間で全単射が実現できることを前提としているが, 実際の回路では定数乗算器やシフタなどがこの前提を満たさない. この場合入出力間で全単射が成り立つように回路要素の機能を拡張することで提案手法が適用できる. 4thIIR では 5 個の演算モジュールのうち 3 個が定数演算器であり, すべての定数乗算器に対して全単射が実現できるように機能の拡張を行った. 実験結果は完全スキャンと比較した場合, 提案方式のハードウェアオーバヘッドはこのような機能の拡張を行っても十分小さいことを示唆している.

2.6. まとめ

本論文では, 与えられたレジスタ転送レベルデータパスに対して, 完全故障検出効率を保証するテスト容易化設計法, および, テスト容易化設計法を適用したデータパス上の回路要素に対するテストプラン生成法を示した. また, 完全故障検出効率を保証するテスト容易化設計法として完全スキャンと提案方式の比較実験を 4 種類の回路に対して行うことで, ハードウェアオーバヘッド, テスト生成時間, テスト系列長の 3 点で, 特に大規模な回路において提案方式が完全スキャン方式より優れていることを示した.

今後の課題としては, データパスに対する条件の緩和が挙げられる. 具体的には,1）異なるビット幅を持つデータ信号線の混在する場合, 2）3 個以上のデータ入力端子や 2 個以上のデータ出力端子を持つ回路要素が存在する場合, にも適用できるようにアルゴリズムを拡張することが考えられる.

	GCD					
	org.		ST		FS	
bit	#FF	#gate	#gate	%OH	#gate	%OH
8	24	259	267	3.1	283	9.3
16	48	529	545	3.0	677	28.0
32	96	873	905	3.7	1161	33.0
	4thIIR					
	org.		ST		FS	
bit	#FF	#gate	#gate	%OH	#gate	%OH
8	96	350	510	45.7	636	81.7
16	192	1192	1512	26.8	1768	48.3
32	384	4316	4641	7.5	5468	26.7
	Paulin					
	org.		ST		FS	
bit	#FF	#gate	#gate	%OH	#gate	%OH
8	56	1152	1184	2.8	1240	7.6
16	112	3848	3912	1.7	4042	5.0
32	224	13848	13976	0.9	14200	2.5
	RISC プロセッサ					
	org.		ST		FS	
bit	#FF	#gate	#gate	%OH	#gate	%OH
32	1280	58158	64497	10.9	67870	16.7

表 2.5 ハードウェアオーバヘッド

第 3 章
テスト容易化高位合成法

3.1. まえがき

半導体技術の発達によって製造可能な論理回路が大規模化している．しかし論理回路の大規模化に伴う設計費用およびテスト費用の増大が問題となっている．

設計費用を削減する目的で高位合成技術が提案されている．高位合成系は設計する回路が実現すべき機能の記述（動作記述）及び回路が満たすべき制約から，動作記述で指定された機能を持ち制約を満たすレジスタ転送レベル（RTL）回路を合成する．動作記述の有する抽象性と記述性の高さが設計工数の削減を可能にしている [2].

高位合成技術は現在も活発に研究が行われており，近年においては面積及び速度等の従来からの制約に加えてテスト容易性に関する制約を同時に満足するテスト容易化高位合成系も提案されている [5, 12]．テスト容易性を考慮せずに合成された RTL 回路に対してテスト容易化設計を施す場合に比べて，テスト容易化高位合成ではより低いハードウエアオーバヘッドで同等のテスト容易性をもつ回路が得られる．また高位合成時に得られた回路に関する情報を利用してテスト生成費用も削減できる．

本論文では演算器の強可検査性 [13] を保証するテスト容易性を制約とするテスト容易化高位合成法について考察する．演算器の強可検査性は，データパス上の任意の演算器に対して外部入力から任意の値を伝達し，その応答を外部出力で観測することが可能であることを保証する．このような条件を満たす回路に対して単一縮退故障集合に対するテスト生成を行った場合，高い故障検出効率が得られる [13].

本論文では以下の 3 つの条件の下で任意の高位合成手続きが生成する RTL 回

路上の演算器が強可検査性を満たすことを示す.

1. データフローグラフが無閉路
2. 全てのレジスタがホールド機能を有する
3. 全演算器で各入力端子毎に出力端子との間に全単射が実現できる

上記の条件は高位合成手続きに関しては制約を持たない．つまり無閉路データフローグラフに対しては，条件（2）,（3）を満たすレジスタや演算器を使用するだけで，既存のあるいは将来提案されるであろう任意の高位合成手続きをそのまま用いて，生成されるデータパスのテスト容易性を向上させることができるという利点を持つ.

また加算器や乗算器，減算器といった通常の設計で用いられる一般的な演算器の多くは条件（3）を満たすので，演算器にはハードウエアオーバヘッドが生じない．従来の高位合成系で生成されるデータパス上の多くのレジスタはホールド機能を有するため，全てのレジスタがホールド機能を持つという条件はハードウエアオーバヘッドをほとんど増加させない.

以下，3.2 節では任意の高位合成系に表れる共通の特徴のみを用いてデータパスの高位合成系をモデル化する．3.3 節では任意の高位合成系で生成されるデータパスがテスト容易となる条件を示し，その条件の下でデータパスがテスト容易となることを 3.3.1 節で証明する．3.4 節では 3.3 節の条件の下で生成されるデータパスのテスト方法について述べ，3.5 節で全体をまとめる.

3.2. 諸定義

一般にデータパスの高位合成系は次の手順で動作記述から RTL データパスを生成する.

1. **フロントエンド:** 動作記述からデータフローグラフを抽出する．データフローグラフは動作記述に表れる演算の依存関係を表した有向グラフで，動作記述上の演算を頂点とする．演算 f の引数となる変数は f に対応するデー

タフローグラフ上の頂点 n_f への流入辺として表され，演算 f の結果を受け入れる変数は n_f の流出辺として表される．

2. **アロケーション:** RTL データパスで使用されるレジスタ及び演算器の種類と個数を決定する．

3. **スケジューリング:** データフローグラフ上の各演算を RTL 回路のクロックに同期した制御ステップに割り当てる．

4. **バインディング:** スケジュール済データフローグラフ上の演算の演算器への割り当て，変数のレジスタへの割り当てを決定し，転送回路でレジスタと演算器を接続する．**転送回路**はある演算が割り当てられた演算器と演算の流入辺及び流出辺にあたる変数が割り当てられたレジスタ間のデータ転送を行うデータパス上の部分回路である．本論文では 2 入力のマルチプレクサを適切に組み合わせて転送回路とする．

本節では本論文で取り上げるバインディング系とその入力であるスケジュール済データフローグラフ，出力であるデータパス及び若干の記号を定義する．

3.2.1 スケジュール済データフローグラフ

問題の簡単化のためにデータフローグラフ上の各演算は 1 または 2 つの入力を持ち 1 出力を持つものと仮定する[8] ．

スケジュール済データフローグラフを，頂点集合 $\mathbf{V}$ と辺集合 $\mathbf{E}$ から成る有向グラフ DFG として次のように定義する．

$$DFG = (\mathbf{V}, \mathbf{E}) \text{ ただし } \begin{cases} \mathbf{V} = \{(op, t_s, t_e) | op \in OP, t_s \in T, t_e \in T\} \\ \mathbf{E} = \{(v_t, v_h, p) | v_t, v_h \in \mathbf{V}, p \in \{0, 1\}\} \end{cases}$$

頂点 $v = (op, t_s, t_e)$ は時刻 t_s に実行を開始し，時刻 t_e に実行を終了する op という種類の演算に対応する．OP は高位合成系で利用可能な演算の集合，T は時刻（自然数）の集合とする．以下では v の演算の種類，実行開始時刻及び実行終

[8] 付録参照.

了時刻をそれぞれ $v.op$，$v.t_s$ 及び $v.t_e$ で表す．$\forall v \in \mathbf{V}(v.t_s \leq v.t_e)$ である．有向辺 $e = (v_t, v_h, p)$ は演算 v_t の結果であり，かつ演算 v_h の p 番目の引数である変数に対応する．以下では e の始点となる演算を $e.v_t$,e の終点となる演算を $e.v_h$, e が $e.v_h$ のどの入力であるかを $e.p$ で表す．

回路外部との入出力は，特別な演算である外部入力演算および外部出力演算として表現する．**外部入力演算**は流出辺のみを持つ頂点で，回路外部から取り込んだ値を流出辺に対応する変数に格納する．**外部出力演算**は唯 1 つの流入辺のみを持つ頂点で，流入辺に対応する変数に格納されている値を回路外部に出力する．また外部入力演算に対応する頂点の流出辺に対応する変数を**外部入力変数**と呼び，外部出力演算に対応する頂点の流入辺に対応する変数を**外部出力変数**と呼ぶ．以下では特に混乱がない限り $\mathbf{V}$ の要素を演算として取り扱い，$\mathbf{E}$ の要素を変数として取り扱う．

次に変数集合 $\mathbf{E}$ 上の 2 項関係 $\rightsquigarrow_u$ および $\rightsquigarrow$ を次のように定める．

$$e_1 \rightsquigarrow_u e_2 \stackrel{\text{def}}{=} (e_1 = e_2 \vee \exists v \in \mathbf{V}(e_1.v_h = v \wedge e_2.v_t = v)$$
$$e_1 \rightsquigarrow e_2 \stackrel{\text{def}}{=} (e_1 \rightsquigarrow_u e_2 \vee \exists e \in \mathbf{E}(e \neq e_1 \wedge e_1 \rightsquigarrow_u e \wedge e \rightsquigarrow e_2)$$

つまり DFG 上で変数 e_1 から e_2 へ至る経路が存在する場合かつその場合に限り $e_1 \rightsquigarrow e_2$ は真である．

本論文では DFG が無閉路であること，つまり次の命題が恒真であることを仮定する．

$$\forall e_1, e_2 \in \mathbf{E}(e_1 \neq e_2 \Rightarrow \overline{e_1 \rightsquigarrow e_2} \vee \overline{e_2 \rightsquigarrow e_1})$$

よって

$$\forall e \in \mathbf{E}(e.v_t.t_e < e.v_h.t_s)$$

である．

DFG 上の各変数 $e \in \mathbf{E}$ に対して関係 $\rightsquigarrow$ を用いて**支配変数集合**を

$$D(e) = \{e' \in \mathbf{E} | e' \rightsquigarrow e\}$$

と定める．

3.2.2 バインディング系

バインディング系は DFG の演算の演算器への割り当て，変数のレジスタへの割り当てを決定し，DFG からデータパス DP を生成する．演算器割り当て，レジスタ割り当てはそれぞれ，DFG の演算集合 $\mathbf{V}$ の分割 $\mathbf{B}_{op} = \{M_1, M_2, \cdots, M_k\}$，および辺集合 $\mathbf{E}$ の分割 $\mathbf{B}_r = \{r_1, r_2, \cdots, r_l\}$ として表される．ここで，$\mathbf{B}_{op} = \{M_1, M_2, \cdots, M_k\}$ は $\mathbf{V}$ の分割なので，

- 各 $i(1 \leq i \leq k)$ について $V_i \neq \phi$
- 各 $i, j(1 \leq i < j \leq k)$ について $V_i \cap V_j = \phi$
- $\bigcup_{i=1}^{k} V_i = \mathbf{V}$

が成り立つ．同様のことが $\mathbf{B}_r$ についても成り立つ．

演算集合 $\mathbf{V}$ の分割 $\mathbf{B}_{op}$ は，各演算集合 $V_i \in \mathbf{B}_{op}$ の演算を同じ演算器に割り当て，異なる演算集合の演算を異なる演算器に割り当てることを表している．変数集合の分割も同様である．このことから，同じ演算集合に属する演算の実行時刻は相異なる必要がある．同様に同じ変数集合に属する変数のうち，始点である演算が異なるものの生存時間は相異なる必要がある．

よって演算の分割 $\mathbf{B}_{op}$ において 1 以上 k 以下の任意の整数 i に対して命題

$$\forall u, v \in M_i(u \neq v \Rightarrow (u.t_e < v.t_s \vee v.t_e < u.t_s))$$

が恒真であり，変数の分割 $\mathbf{B}_r$ において 1 以上 l 以下の任意の整数 i に対して次の命題が恒真であると仮定する．

$$\forall e, e' \in E_i(e.v_t \neq e'.v_t \Rightarrow e.v_t.t_e < e'.v_h.t_s \vee e'.v_t.t_e < e.v_h.t_s)$$

以下では $M_i \in \mathbf{B}_{op}$ なる演算集合 M_i と M_i 上の演算が割り当てられる演算器を同一視する．また変数集合 $r_i \in \mathbf{B}_r$ と r_i 上の変数が割り当てられるレジスタを同一視する．さらに $\mathbf{B}_{op}$ をデータパス上の演算器集合と同一視し，$\mathbf{B}_r$ をデータパス上のレジスタ集合と同一視する．ここで，2 入力演算器 M に割り当てられている全ての 2 入力演算について 2 つの入力変数の始点が同じ演算である場合，M に

割り当てられている 2 入力演算は 1 入力演算に変換することができる．このような場合，M に割り当てられている全ての演算を 1 入力演算に変換し，M に対応する演算器も 1 入力とする．

次にデータパス $DP = (\mathbf{B}_{op}, \mathbf{B}_r, MMX)$ を次のように定義する．

$\mathbf{B}_{op}$ はデータパス上の演算器集合,$\mathbf{B}_r$ はデータパス上のレジスタ集合であり，MMX は転送回路である．

本論文では簡単のためデータパス上に現れるすべての**回路要素**（外部入力，外部出力，演算器，レジスタ及びマルチプレクサ）のデータ入力端子及びデータ出力端子の定義域及び値域が等しいものとする．以下ではデータ入力端子 x の定義域を $DOM(x)$, データ出力端子 z の値域を $RNG(z)$ とする．

転送回路は 2 入力のマルチプレクサから成るフィードバックを持たないデータパス上の部分回路で，$\mathbf{B}_{op}$ に含まれる全ての演算器と $\mathbf{B}_r$ に含まれる全てのレジスタの出力を入力とし，$\mathbf{B}_{op}$ に含まれる全ての演算器と $\mathbf{B}_r$ に含まれる全てのレジスタの入力を出力とする．

$v.t_s = t$ である演算 v に対して v が割り当てられてる演算器を M，$e.p = 0$ および $e.p = 1$ となる v の流入辺 e（変数）が割り当てられているレジスタをそれぞれ r_0，r_1 とする．MMX 上に r_0 および r_1 から M に至る経路が存在して，時刻 t においてこれらの経路に沿って任意の値が伝達されるように各マルチプレクサが制御されるものとする．同様に $v'.t_e = t$ である演算 v' に対して v' が割り当てられてる演算器を M'，v' の流出辺 (変数) が割り当てられているレジスタを r' とすると，MMX 上に M' から r' に至る経路が存在して，時刻 t においてこれらの経路に沿って任意の値が伝達されるように各マルチプレクサが制御されるものとする．

次にデータパス上の経路 $p = \cdots, r_i, M_i, r_{i+1}, \cdots$ を演算器とレジスタが交互に現れる系列として定義する．ただし p 上で隣接関係にある任意のレジスタ r_i, r_{i+1} および演算器 M_i から成る組に対してつぎの 2 つの条件が成り立つものとする．

$$\exists u \in M_i(\exists e \in r_i(e.v_h = u)) \text{ および } \exists v \in M_i(\exists e' \in r_{i+1}(e'.v_t = v))$$

これらの条件は，MMX 上に r_i から M_i へ至る経路と M_i から r_{i+1} へ至る経路が存在することを保証する．

さらにデータパス上の任意の回路要素 M において M がコントローラと直接接続している入力端子（制御端子）は回路外部から任意の時刻で任意の値に設定可能であり M がコントローラと直接接続している出力端子（状態端子）は任意の時刻で回路外部から観測可能であるものと仮定する．

3.3. DP のテスト容易性に関する十分条件

本論文ではデータパス DP のテスト容易性として,「DP 上の各演算器が強可検査 [13] である」という条件を用いる．ここで，回路要素 M が以下の 2 つの条件を満たすとき M が強可検査であると言う．

- **強可制御性:** 外部入力から任意の値の組を M の 2 つのデータ入力端子まで伝達可能
- **強可観測性:** M のデータ出力端子に表れる任意の値を外部出力まで伝達可能

定理 1 テスト容易データパス生成の十分条件
以下の 3 条件の下で，任意のバインディング系によって DFG から生成される任意の DP 上の任意の演算器が強可検査である．

- DFG が無閉路
- 全てのレジスタがホールド機能を有する
- 演算器において各入力端子と出力端子の間に全単射が成り立つ

証明: 次節の補題 *2* 及び補題 *3* より証明される． □

定理 1 において「演算器の各入力端子と出力端子の間に全単射が成り立つ」とは，演算器が実現する関数族にある関数 f(または g) が存在して次の条件を満たすことである．

- **演算器が 1 入力の場合:** データ入力端子を x, データ出力端子を z とした場合， 次式が恒真．

$$\forall u, v \in DOM(x)(u \neq v \Leftrightarrow f(u) \neq f(v))$$

- **演算器が 2 入力の場合:** データ入力端子を x, y, データ出力端子を z とした場合，次の 2 つの式が恒真.

$$\forall u, v \in DOM(x)(\exists a \in DOM(y)(u \neq v \Leftrightarrow f(u,a) \neq f(v,a)))$$

$$\forall u, v \in DOM(y)(\exists a \in DOM(x)(u \neq v \Leftrightarrow g(a,u) \neq g(a,v)))$$

つまり，1 入力演算器では入出力間の全単射 f を実現できることを意味する．また 2 入力演算器では，一方の入力に定数を印加（例えば加算器の場合，定数 0 を印加）することで，他方の入力と出力の間に全単射 f(または g) が実現できることを意味する．この全単射 f（または g）を用いれば，演算器の出力を 1 つの入力端子で制御可能であり，また演算器の入力を出力端子で観測可能である．以下では関数 f（または g）を演算器 M の**全単射関数**と呼ぶ.

次節では定理 1 の条件の元で任意の演算器が強可検査であることを示す.

3.3.1 証明

定義 1 レジスタの依存関係
レジスタ集合 $\mathbf{B}_r$ 上の 2 項関係 $Rdep()$ を次のように定義する.

$$Rdep(r_1, r_2) \stackrel{\text{def}}{=} (\forall e \in r_2(\exists e' \in r_1(e' \in D(e))))$$

□

補題 1 レジスタの強可制御性
DP 上の全てのレジスタに任意の値が設定可能である.
証明: 任意の変数 e について，その変数の支配変数集合に含まれる変数のみを用いて，任意の値が設定可能である．また定義 *1* より，$Rdep(r_1, r_2)$ が偽の場合，ある変数 e が r_2 に割り当てられていて r_1 に割り当てられているどの変数も $D(e)$ に含まれない．よって r_1 に割り当てられているどの変数も変化させることなく e に任意の値を設定することができる．このことは $Rdep(r_1, r_2)$ が偽であるならば r_1 をホールドしたままでも外部入力から r_2 に任意の値を設定できることを意味する.

また DFG が無閉路なので次の式が常に成り立つ.

$$\forall r_1, r_2 \in \mathbf{B}_r(\overline{Rdep(r_1, r_2)} \vee \overline{Rdep(r_2, r_1)})$$

また $Rdep()$ の定義より次の式も常に成り立つ.

$$\forall r_1, r_2, r_3 \in \mathbf{B}_r(Rdep(r_1, r_2) \wedge Rdep(r_2, r_3) \Rightarrow Rdep(r_1, r_3))$$

よって $Rdep(r_1, r_2) \Rightarrow r_2 \geq r_1$ であるように r_1, r_2 の大小関係 $\geq$ を定めると，$\geq$ は $\mathbf{B}_r$ 上の半順序関係を成す．ゆえに $\geq$ を拡張して得られる任意の全順序関係において，最も大きいレジスタ r から順に r に値を設定してから r をホールドすることで，DP 上の全てのレジスタに任意の値を設定することができる． □

補題 2 演算器の強可制御性
DP 上の任意の演算器 M は強可制御である.
証明: 転送回路 MMX の定義から M の各入力端子に対して，値を供給するレジスタ r_1, r_2(ただし $r_1 \neq r_2$) が存在する．補題 1 より全レジスタに対して任意の値が設定できるので，r_1, r_2 に値を設定した後に，MMX に対して適当な制御信号を与えることで，任意の値を M の入力端子に伝達することができる．M が *1* 入力の場合も，同様に証明できる． □

定義 2 レジスタ間の隣接関係
DP 上の 2 入力演算器 M に対して，レジスタ集合 $\mathbf{B}_r$ 上の 3 項関係 $Adj_M()$ を次のように定義する.

$$\begin{aligned} Adj_M(r_1, r_2, r_3) \stackrel{\text{def}}{=} \; & \exists e_1 \in r_1, \exists e_2 \in r_2, \exists e_3 \in r_3 \\ & (e_1.v_h \in \mathbf{M} \wedge e_2.v_h \in \mathbf{M} \wedge e_3.v_t \in \mathbf{M} \wedge e_1.p = 1 - (e_2.p)) \end{aligned}$$

□

つまり $Adj_M(r_1, r_2, r_3)$ が真であれば演算器 M の 2 つのデータ入力端子にレジスタ r_1, r_2 がそれぞれ転送回路 MMX を通して接続しており，M の出力端子にレジスタ r_3 が MMX を通して接続している.

定義 3 隣接レジスタ間の伝達関係
演算器 M を介して隣接するレジスタ間の関係 $store_M()$ を次のように定義する．

- **演算器 M が 1 入力の場合:**

$$store_M(r_1, r_2) \stackrel{\text{def}}{=} \exists e \in r_1(\exists e' \in r_2(e.v_h \in M \wedge e'.v_t \in M))$$

- **演算器 M が 2 入力の場合:**

$$store_M(r_1, r_2) \stackrel{\text{def}}{=} \exists r_3 \in \mathbf{B}_r(Adj_M(r_1, r_3, r_2) \wedge \overline{Rdep(r_1, r_3)}))$$

□

M が 2 入力の場合，$store_M(r_1, r_2)$ が真であれば r_1 をホールドしたままで r_3 に任意の値を設定することができる．そこで r_1 をホールドしたまま r_3 に r_1–r_2 間で全単射を実現するために必要な値を設定した後で M が全単射関数を実現するような制御信号を与えることで r_1 上の任意の値を M を介して r_2 に伝達することができる．

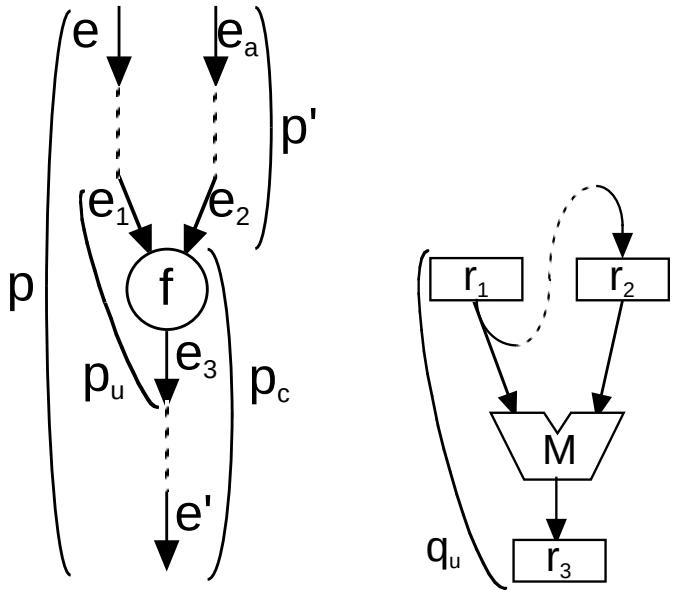
M が 1 入力の場合は M が全単射関数を実現するような制御信号を与えるだけで r_1 上の任意の値を M を介して r_2 に伝達することができる．

定義 4 レジスタ間の伝達関係
レジスタ集合 $\mathbf{B}_r$ 上の 2 項関係 $store^*()$ を次のように定義する．

$$store^*(r_1, r_2) \stackrel{\text{def}}{=} \exists M \in \mathbf{B}_{op}(store_M(r_1, r_2) \vee \exists r \in \mathbf{B}_r(store^*(r_1, r) \wedge store_M(r, r_2)))$$

□

$store^*(r_1, r_2)$ が真であれば，DP 上にある経路 $p = r_1, \cdots, r_2$ が存在して p 上の任意の部分経路 r, M, r' において $store_M(r, r')$ が真である．よって各部分経路毎に r から r' へ値を伝達できるので，r_1 の値を r_2 に伝達可能である．



(a) DFG 上の経路 *(ただし* $e_a \in r_1$*)*　　*(b)* q_u 回りの回路要素

図 3.1 演算 f と対応する演算器 M

補題 3 命題（従属伝達関係）の恒真性
次の命題は恒真.

$$\forall r, r' \in \mathbf{B}_r(Rdep(r, r') \Rightarrow store^*(r, r'))$$

証明: 命題の否定 $Rdep(r, r') \wedge \overline{store^*(r, r')}$ が真であると仮定する.

$Rdep(r, r')$ より $e \rightsquigarrow e'$ をみたす変数の組 (e, e') （ただし，$e \in r, e' \in r'$）が存在する．ここで e を始点，e' を終点とする DFG 上の経路を p,p に対応する DP 上の経路を q とする．すなわち $p = a_1, a_2, \cdots, a_n$, $q = b_1, b_2, \cdots, b_n$ において各整数 $i(1 \leq i \leq n)$ に対して $a_i \in b_i$ である.

$\overline{store^*(r, r')}$ よりレジスタ r が保持している値を経路 q に沿ってレジスタ r' まで伝達することはできない．よって q 上に部分経路 r_1, M, r_3 が少なくとも *1* つ存在して，$\overline{store^*(r_1, r_3)}$ である．このような条件を満たす q 上の部分経路のうち対応する DFG 上の部分経路 p_u($p_u = e_1, f, e_3$ とする）（図 *3.1-*（*a*））が p の終点に最

も近いものを q_u($q_u = r_1, M, r_3$ とする）とする（図 *3.1-*（*b*））．このとき，p_u 上の演算 f を始点とし，p の終点 e' を終点とする p 上の部分経路を p_c とし，p_c に対応する q 上の部分経路を q_c とすると，q_c の始点である演算器の出力を q_c の終点であるレジスタ r' まで伝達できることを意味する．

ここで DFG 上の f を終点とする e_1 以外の変数を e_2，e_2 が割り当てられているレジスタを r_2 とする．このとき $\overline{store_M(r_1, r_3)}$ より $Rdep(r_1, r_2)$ が真．よって $\exists e_a \in r_1(e_a \rightsquigarrow e_2)$ も真である．

DFG 上の e_a を始点，e_2 を終点とする経路を p' とする．DFG が無閉路であることより，p' は p_c と共通部分を持たない[9] ．また $store_M(r_2, r_3)$ より，e_2 の値は e_3 に伝達できる．従って， $e_2, f, e_3, \cdots, e'$ で任意の値を伝達できる．

また p' に対して回路要素 r'_1, r'_2, r'_3, M' 及び p' 上の部分経路 p'_c を p に対する回路要素 r_1, r_2, r_3, M 及び部分経路 p_c と同様に定めることができる．

ここで p'_c を p に連結した経路を新たに p_c とし，r'_1, r'_2 を新たに r_1, r_2 とすると，上記の手続きを繰り返すことで任意の p_c に対してその経路上に現れる演算の個数がいくらでも大きな経路を生成することができる．これはデータフローグラフが有限であることに矛盾する．よって補題は示された． □

補題 4 命題（隣接伝達関係）の恒真性
次の命題は恒真.

$$\forall M \in \mathbf{B}_{op}(\forall r_1, r_2, r_3 \in \mathbf{B}_r(Adj_M(r_1, r_2, r_3) \Rightarrow store^*(r_1, r_3) \wedge store^*(r_2, r_3)))$$

証明: $Adj_M(r_1, r_2, r_3)$ が真の場合を考える．まず DFG が無閉路であることから命題 $\forall r_1, r_2 \in \mathbf{B}_r(\overline{Rdep(r_1, r_2)} \vee \overline{Rdep(r_2, r_1)})$ は真．ここで $Rdep(r_1, r_2)$ と $Rdep(r_2, r_1)$ の真偽について以下の *2* 通りに場合分けが可能.

- $Rdep(r_1, r_2)$ と $Rdep(r_2, r_1)$ が共に偽の場合*:*
  2 項関係 $store_M()$ の定義より $store_M(r_1, r_3)$ 及び $store_M(r_2, r_3)$ は共に真である．よって命題は真

[9] p' の終点は e_2 であり p_c の始点は f である．p と p' が共有点 o（演算でも変数でも良い）を持ったとすると p' 上で o から e_2 へ至る経路が存在し，p 上で f から o へ至る経路が存在する．ここで $e_2.v_h = f$ であるので o から o へ至る経路が DFG 上に存在することになるがこれは DFG が無閉路という仮定に反する．

- $Rdep(r_1, r_2)$; $Rdep(r_2, r_1)$ の一方が真で他方が偽の場合:
  一般性を失うことなく $Rdep(r_1, r_2)$ が偽, $Rdep(r_2, r_1)$ が真であると仮定する. *2*項関係 $store_M()$ の定義より $store_M(r_1, r_3)$ は真である. よって $store^*(r_1, r_3)$ は真.

  また補題 *3* より, $Rdep(r_2, r_1) \Rightarrow store^*(r_2, r_1)$ が成り立つので, $store^*(r_2, r_1)$ は真. $store_M(r_1, r_3)$ が真であることより $store^*(r_2, r_3)$ も真である. よって命題は真.

以上より補題は成り立つ. □

補題 5 演算器の強可観測性
DP 上の任意の演算器 M は強可観測である.
証明: 転送回路 MMX の定義から, M の出力端子の値を入力とするレジスタ r が存在する. DFG に関する仮定より r を始点, ある出力変数が割り当てられているレジスタ r' を終点とする DP 上の単純経路 p が DP 上に存在する. ここで補題 *4* より, 次の命題は真である.

$$\forall M \in \mathbf{B}_{op}(\forall r_1, r_2, r_3 \in \mathbf{B}_r(Adj_M(r_1, r_2, r_3) \Rightarrow store^*(r_1, r_3) \wedge store^*(r_2, r_3)))$$

よって p 上の任意の演算器 M に対してその前後にあるレジスタを含んだ p 上の任意の部分経路 r, M, r' において r から r' へ値が伝達可能であるので補題は成り立つ. □

3.4. テストプラン生成手続き

前節で示した条件の下で任意の高位合成系を用いて合成されたデータパス上の各回路要素を適切に制御することで, 外部入力から任意の値の組を演算器の 2 つのデータ入力端子まで伝達することが可能であり, また演算器のデータ出力端子に表れる任意の値を外部出力まで伝達することも可能である. このときに必要な制御信号の時系列（テストプラン）は, 次のように容易に生成できる.

補題 1,2 の条件に従って任意の演算器を強可制御とするデータパスへの制御信号の時系列が生成できる．また補題 3 の証明から $Rdep(r, r')$ であるようなレジスタの組 (r, r') に対して r を始点 r' を終点とする DP 上のある経路 q が存在して，q 上の任意の部分経路 r_1, M, r_2 において $store_M(r_1, r_2)$ であることがわかる．よって，$store_M(r_1, r_2)$ が真となる経路 r_1, M, r_2 のみを用いて外部出力変数が割り当てられているレジスタを根とする DP 上の林 G を生成することができて，この林には DP 上の全てのレジスタが含まれる．そこで任意のレジスタに対して外部出力変数が割り当てられているレジスタに至る G 上の経路を選択することができて，この経路上の各演算器において定義 3 に従ってデータパスを制御することで任意のレジスタの値をデータパスの外部出力まで伝達することができる．

3.5. まとめ

本論文では強可検査性に基づくテスト容易性を満たすデータパスを任意の高位合成手続きで生成するための条件を提案した．提案した条件に基づいて生成されるデータパスはテスト容易であるだけでなくテスト容易性の改善に伴うハードウエアオーバヘッドが小さい．また高位合成手続きそのものに対しては何らの制約を課さないので従来の高位合成手続きや将来にわたって提案される種々の特徴を持った高位合成手続きに対して，それらに何らの変更を加えることなく生成されるデータパスのテスト容易性を改善することができる．

今後の課題としては，閉路を持つデータフローグラフへの対応等が挙げられる．

第 4 章
結論

論理回路の大規模化に伴って論理回路の設計とテストに要する費用の増大が問題になっている．設計に要する費用を削減する目的で論理合成系を用いてレジスタ転送レベル回路から論理回路を生成する手法が利用されている．また高位合成技術を利用して動作記述からレジスタ転送レベル回路を生成することで設計費用の更なる低減を図ることが提案されている．

製造された論理回路の信頼性を保証する目的で高い故障検出効率を持つテスト系列を用いたテスト実行が必要であるが，一般の順序論理回路に対して高い故障検出効率を持つテスト系列を生成することは困難である．そこでテスト生成が容易になるように順序回路に対して設計を追加するテスト容易化設計法が提案されているが，設計の追加に伴うハードウエアオーバヘッドが生じるという問題点がある．

テスト容易性を確保しつつハードウエアオーバヘッドが抑制可能なテスト容易化設計法を実現するために論理回路よりも抽象度の高いレジスタ転送レベル回路やさらに抽象度の高い動作記述から得られる情報を利用することが提案されている．

本論文では対応する論理回路がテスト容易となるためのレジスタ転送レベルデータパスが満たすべき条件として強可検査性を提案した．また与えられたレジスタ転送レベルデータパスを強可検査なデータパスに変換するテスト容易化設計法を提案した．提案したテスト容易化設計法と現在一般的に利用されている論理回路を対象としたテスト容易化設計法である完全スキャン設計法を実験により比較し，提案手法は従来手法と同等以上のテスト容易性を確保しつつハードウエアオーバヘッドおよびテスト実行時間を削減することを示した．さらに演算器が強可検査であるレジスタ転送レベルデータパスを生成するテスト容易化高位合成を

提案し，任意の高位合成手続きで生成されるデータパス上の演算器の強可検査性が保証されるための条件を示した．

本論文での提案を含めて，様々なテスト容易化設計法およびテスト容易化高位合成法が提案されている．しかしながら半導体製造技術の発達を背景として論理回路の大規模高性能化が引き続き進展するものと考えられる．よってテスト容易化設計およびテスト容易化高位合成に関する研究の継続を通して製品として出荷される論理回路の信頼性向上とテスト費用削減のための努力が引き続き行われる必要があると考えられる．

付録

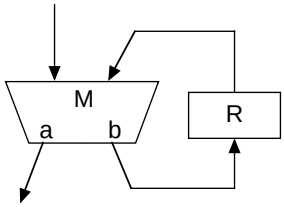
A. 演算に対する条件の緩和

3.2 節で導入したデータフローグラフでは、各演算は 1 または 2 個の入力を持ち 1 出力を持つものと仮定した。演算がこの仮定を満たさない場合、つまり入力が 3 個以上の演算あるいは出力が 2 個以上の演算について、本論文で提案したテスト容易性を満たすための条件を述べる。

A.1 入力が 3 個以上の演算

n 入力 1 出力の演算 f が n 入力 1 出力の演算器 M に割り当てられている場合を考える。マルチプレクサのみを介して M の入力に接続しているレジスタの集合を $\mathbf{R}_{in}$ とする。$\mathbf{R}_{in}$ 中の任意のレジスタの値を M の出力に伝達可能ならば、定理 1 が成立する。

$\mathbf{R}_{in}$ は補題 1 で定義した大小関係の下で半順序を成すので $\mathbf{R}_{in}$ の極大な要素からなる集合を $\mathbf{R}_{max}$ とする。補題 5 と同様に $\mathbf{R}_{in}$ 中の任意のレジスタの値を $\mathbf{R}_{max}$ 中のいずれかのレジスタに伝達可能であり、かつ補題 1 と同様に $\mathbf{R}_{max}$ の要素であるレジスタ r の値を保持したまま $\mathbf{R}_{in} - \{r\}$ 中の全てのレジスタの値を任意の値に変更可能である。以上より多入力演算 f を有するデータフローグラフで定理 1 が成立する十分条件は、M の入力端子毎に残りの入力端子に適当な定数を印加することで出力端子との間に全単射を実現する関数が、M が実現する関数群に含まれることである。



※ M の出力 b の値は M を介して出力 a に伝達

図 4.1 多出力演算器 M が強可検査とならない例

A.2 出力が２個以上の演算

多出力の演算器 M を用いてデータパスを構成した場合、M の応答を M を介して観測することしかできない事がある (図 4.1)。この場合 M は強可検査とはならない [13]。またデータパスを構成する全ての回路要素が 1 出力であればテスト対象の回路要素を介した応答の観測を要しないので、全単射関数とホールド機能を適切なレジスタ及び演算器に割り当てることでデータパスを強可検査とすることができる。よって高位合成によって得られるデータパスが１出力の回路要素のみを用いて構成できればよい。

よってデータパスを１出力の回路要素のみを用いて構成する目的で、m 入力 n 出力の演算 f を m 入力 1 出力の演算 $f_1, \cdots, f_n$ で置き換え、個々の１出力関数を割り当て可能な１出力演算器を高位合成時に用いる回路要素ライブラリに登録すると、定理１が成立する。

謝辞

本研究の機会を与えて下さるとともに，本研究の全過程を通じて，方針や問題点などのあらゆる面において，懇切丁寧な御指導と御助言を賜わりました藤原秀雄教授に心から感謝の意を表します.

本研究を進めるにあたり，御指導と御助言を頂きました渡邉勝正教授ならびに福田晃教授に感謝の意を表します.

本研究の全過程を通じて，貴重な御討論，御助言を頂き，懇切丁寧に直接的な御指導を頂きました大阪大学基礎工学部の増澤利光教授に深く感謝致します.

本研究の全過程を通じて，貴重な御討論，御助言を頂き，懇切丁寧に直接的な御指導を頂きました井上美智子助教授ならびに広島市立大学情報機械システム工学科の井上智生助教授に深く感謝致します.

本研究の全過程を通じて，貴重な御討論，御助言を頂き，懇切丁寧に直接的な御指導を頂きました大竹哲史助手に深く感謝致します.

本研究を進めるにあたり，貴重な御討論，御助言を頂きました ウィスコンシン大学（米国）計算機工学科 の Kewal K. Saluja 教授,（株）半導体理工学研究センター（STARC）の小澤時典取締役研究推進部長,（株）日立製作所中央研究所の畠山一実氏，ソニー（株）セミコンダクタカンパニーシステム LSI 部門の小野寺岳志氏ならびに三洋電機（株）研究開発本部マイクロエレクトロニクス研究所の小椋功氏に感謝致します.

本研究における実験に用いた情報科学研究科 CAD システムの保守管理をして頂くとともに，その使用方法について丁寧な御指導および御助言を頂きました木村晋二助教授に深く感謝致します.

また，日頃より有益な御討論，御援助を頂きました情報論理学講座の諸氏に感謝致します.

参考文献

[1] H.Fujiwara, "Logic testing and design for testability," The MIT press, Cambridge, 1985

[2] P. Michel, U. Launther and P. Duzy :"The Synthesis Approach to Digital System Design, " Kluwer academic publishers group, Dordrecht, 1992

[3] B.T.Murray and J.H.Hayes: "Hierarchical test generation using pre computed tests for modules," IEEE Trans. on CAD, vol. 9 no.6,pp.594-603,June 1990.

[4] J.Lee and J.H.Patel, "Hierarchical test generation under architectural level functional constraints," IEEE Trans. on CAD, vol.15, no.9, pp.1144-1151, Sep. 1996

[5] S.Bhatia and N.K.Jha:"Genesis: A behavioral synthesis system for hierarchical testability," Proc. European Design and Test Conference , pp272-276 , 1994

[6] I.Ghosh,A.Raghunath and N.K.Jha:"Design for hierarchical testability of RTL circuits obtained by behavioral synthesis," Proc. 1995 IEEE Int. Conf. on Computer Design ,pp.173-179 , 1995

[7] I.Ghosh,A.Raghunath and N.K.Jha:"A design for testability techinique for RTL circuits using control/dataflow extraction" Proc. 1996 IEEE/ACM Int.Conf.on CAD ,pp.329-336,1996,

[8] I.Ghosh, A.Raghunathan and N.K.Jha :"Design for hierarchical testability of RTL circuits obtained by behavioral synthesis,"IEEE Trans. on CAD,Vol. 16,No. 9,pp. 1001-1014,Sept. 1997

[9] I.Ghosh, A.Raghunathan and N.K.Jha:“Hierarchical test generation and design for testability methods for ASPP’s and ASIP’s,” IEEE Trans. on CAD, Vol.18,No.3,pp.357-370,March 1999.

[10] R.B.Norwood and E.J.McCluskey:“Orthogonal scan: Low overhead scan for data paths”, Proc. 1996 Int. Test Conf. ,pp.659-668 ,1996

[11] R.B.Norwood and E.J.McCluskey:“High-level synthesis for orthogonal scan,” Proc. 15th VLSI Test Symp, pp370-375 ,1997

[12] M.Inoue, T.Higashimura, K.Noda, T.Masuzawa and H.Fujiwara :”A high-level synthesis method for weakly testable data paths,” Proc. IEEE 7th Asian Tst Symposium, pp. 44-45, 1998

[13] 和田弘樹，増澤利光，K.K.Saluja，藤原秀雄:”完全故障検出効率を保証するレジスタ転送レベルデータパスの非スキャンテスト容易化設計法，”電子情報通信学会論文誌，Vol. J82-D-I，No. 7，pp.843-851，Jul. 1999

[14] S. Ohtake, H. Wada, T. Masuzawa and H. Fujiwara: ”A Non-Scan DFT Method at Register-Transfer Level to Achieve Complete Fault Efficiency,” Proc. Asia and South Pacific Design Automation Conference 2000 (ASP-DAC2000), pp.599-604, Jan. 2000 .