NAIST-IS-DD0861201

博士論文

大域非同期局所同期システムにおける テスト品質向上に関する研究

岩田 大志

2011年3月1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。

岩田 大志

審査委員：

藤原 秀雄 教授	（主指導教員）
中島 康彦 教授	（副指導教員）
井上 美智子 准教授	（副指導教員）
大竹 哲史 助教	（副指導教員）

大域非同期局所同期システムにおける テスト品質向上に関する研究*

岩田 大志

内容梗概

大域非同期局所同期 (GALS) 設計は，多機能化，大規模化，高速化する VLSI の設計で直面する様々な問題を解決する現実的な手段と考えられている．一方で，VLSI のテストは製品の信頼性を保証するために必要であるが，テスト容易化設計なしに正常に動作する製品のみを選別する，高いテスト品質を達成することはできない．同期式回路の代表的なテスト容易化設計方法に，回路中のフリップフロップを外部から制御可能なスキャン素子に設計変更する完全スキャン設計がある．完全スキャン設計を適用した回路の組合せ回路に対しては任意のテストパターンが印加できるが、通常動作では起こりえない動作をすることがあり正しく動作する回路を不良と判断してしまう過剰テストが行われていると考えられている．特に遅延故障のテストの際には，遷移が伝搬しないパス（フォールスパス）が数多く活性化し，その場合タイミング違反による過剰テストが多発する．フォールスパスを判定することで過剰テストを緩和できるが，多数のパスが存在する大規模なゲートレベル回路に対してフォールスパスを判定することは現実的でない．本論文ではより高位の設計情報を利用し，レジスタ転送レベル (RTL) でフォールスパスを判定し，RTL フォールスパスをゲートレベルで利用する手法を提案する．従来の手法では，論理合成に現実的でない制約を置いていたが，本論文では，論理合成に制約を置かない場合にもパスマッピングを実現するパスマッピング法を提案し，さらに，その制約を緩和する新しい論理合成手法を提案した．実験結果

*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-DD0861201, 2011 年 3 月 1 日.

では，提案したパスマッピング法を用いて多数のゲートレベルフォールスパスが実用的な時間で得られることを示した．

同期式回路のテスト環境を非同期回路に応用するために，非同期式回路に対する完全スキャン設計が提案されているが，従来のテスト手法では組合せ回路，順序素子の両方に対して，検出可能な故障をすべて検出する完全なテストが保証されない．本論文では組合せ回路部分に対して完全なテストを保証する完全スキャン設計を提案し，順序素子の完全なテストを保証する新しいスキャン素子を提案した．実験結果では，提案したスキャン素子が完全なテストを保証しながら従来のスキャン素子と同等のオーバーヘッドで実現できることを示した．

本論文で提案した手法を用いることで，同期式回路に対する過剰テストの緩和と，非同期式回路に対するテスト不足を解消することができるため，GALS システムのテスト品質を向上させる事が可能となる．

キーワード

大域非同期局所同期システム，パスマッピング，フォールスパス，非同期式回路のテスト，2 部完全スキャン設計，スキャン C 素子

Studies on Improvement in Test Quality for Globally Asynchronous Locally Synchronous Systems*

Hiroshi Iwata

Abstract

Globally asynchronous, locally synchronous (GALS) design has been known as a realistic hardware design solution for many difficulties due to the continuous scaling of semiconductor technology. On the other hand, by using the current test techniques, the test quality for both of the synchronous part and the asynchronous part in the circuits is not sufficient. For testing the synchronous part, the full scan design which changes all the sequential elements in the circuit to fully controllable and observable scan elements, is used to reduce the test complexity. Though the testability of the full scan design is that any test pattern can be applied to the combinational part and any test response of the part can be captured, some faults that do not affect the normal operation can be detected. The situation is called over-testing. Therefore, it is needed to extract only the circuits which can be operated correctly to improve the test quality. Especially, for testing path delay faults, the number of over-testing is extremely large since there are many false paths in the circuit. Though over-testing can be alleviated by identifying the false paths, identification of the false paths at gate level is not practical. To handle the false path on the high level design, we propose two path mapping methods

*Doctoral Dissertation, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0861201, March 1, 2011.

which propagate the RTL false path information to gate level. Until now, to map the RTL false paths to the corresponding gate level false paths, there exists an impractical assumption on the logic synthesis. In this dissertation, we propose a method for mapping RTL false paths to their corresponding gate level paths without such a specific logic synthesis and a synthesis method that alleviates the impact of the restriction. In experimental results, many gate level false paths were obtained by the proposed methods in practical time.

For testing the asynchronous part, several full scan design methods for asynchronous circuit that can be tested by using the same environments of synchronous one, have been proposed. However, these methods cannot guarantee complete test for both of the combinational part and the sequential part where complete test means that all the detectable faults in the circuit are detected by the test. Therefore, by using the current test techniques, the test quality for asynchronous circuits is not sufficient. In this dissertation, we propose a full scan design method which guarantees complete test for the combinational part and a new scannable element which guarantees complete test for all the sequential elements on scan paths. In experimental results, proposed scannable element was implemented with low area and performance overhead comparable to the previous best method in terms of overhead.

By using the proposed methods in this dissertation, test quality for GALS system is improved since over-testing is alleviated for synchronous circuits and complete test for asynchronous circuits is guaranteed.

Keywords:

Globally asynchronous locally synchronous system, path mapping, false path, asynchronous circuit testing, bipartite full scan design, scannable C-element

業績リスト

論文誌

1. Hiroshi Iwata, Satoshi Ohtake and Hideo Fujiwara: “A Method of Path Mapping from RTL to Gate Level and Its Application to False Path Identification,” IEICE Transactions on Information and Systems, Vol. E93-D, No. 7, pp. 1857-1865, July 2010.

査読付き国際会議

1. Hiroshi Iwata, Satoshi Ohtake, Michiko Inoue and Hideo Fujiwara: “Bipartite full scan design: a DFT method for asynchronous circuits,” IEEE 19th Asian Test Symposium, pp.206-211, Dec. 2010.

2. Michiko Inoue, Akira Taketani, Tomokazu Yoneda, Hiroshi Iwata and Hideo Fujiwara: “Test Pattern Selection to Optimize Delay Test Quality with a Limited Size of Test Set,” 15th IEEE European Test Symposium, pp.260, May 2010.

3. Satoshi Ohtake, Hiroshi Iwata and Hideo Fujiwara: “A synthesis method to propagate false path information from RTL to gate level,” The IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems 2010, pp.197-200, April 2010.

4. Hiroshi Iwata, Satoshi Ohtake and Hideo Fujiwara: “Enabling False Path Identification from RTL for Reducing Design and Test Futileness,” The 5th

IEEE International Symposium on Electronic Design, Test and Applications, pp.20-25, January 2010.

5. Michiko Inoue, Akira Taketani, Tomokazu Yoneda, Hiroshi Iwata and Hideo Fujiwara: "Optimizing Delay Quality with a Limited Size of Test Set," Proceedings of IEEE International Workshop on Reliability Aware System Design and Test, pp.46-51, Jan. 2010.

6. Hiroshi Iwata, Satoshi Ohtake and Hideo Fujiwara: "An approach to RTL-GL path mapping based on functional equivalence," 9th IEEE Workshop on RTL and High Level Testing, pp. 63-68, November 2008.

テクニカルレポート

1. 岩田大志, 大竹哲史, 井上美智子, 藤原秀雄: "C 素子スキャンパスを用いた非同期式順序回路に対する完全スキャン設計法," 信学技報 (DC2010-8), Vol. 110, No.106, pp. 1-6, 2010.

2. 岩田大志, 大竹哲史, 藤原秀雄: "機能等価性情報を用いた RTL-GL パスマッピングの一手法," 信学技報 (VLD2008-34), No.107, pp. 13-18, 2008.

出願特許

1. 大竹哲史, 岩田大志, 井上美智子: "スキャン C 素子およびそれを備えた半導体集積回路ならびにその設計方法およびテストパターン生成方法," 特願 2010-138609, 2010 年 6 月 17 日 (出願中).

目 次

図 目 次

表 目 次

第1章　緒論

1.1. 研究背景

現在，あらゆるコンピュータで利用されている VLSI(Very Large Scale Integration) は広く普及している．近年の半導体技術の進歩に伴い，VLSI は多機能化，大規模化，高速化しており，VLSI の効率的な設計が求められている．効率的に VLSI を設計するために，あらかじめ機能単位でハードウェアを設計し，その機能単位を再利用することや，設計資産 (IP) として開発された機能単位を購入することで，VLSI の設計を行っている．この機能単位をコア，モジュールと呼ぶが，この複数のコアを 1 つのチップに集約し，それらを接続して VLSI の設計を行うシステムオンチップ (SoC) やネットワークオンチップ (NoC) が多く開発されている．

大規模で高速化を続ける SoC や NoC を開発するにあたり，特に問題となるのはクロックの分配である．一般にデジタル回路はクロックに同期して回路が動作を行う同期式設計を用いて設計されるが，このクロックを大規模回路全体に分配することは，電力，タイミングのずれなどの問題により，非常に困難である．この問題を解決するために，回路をクロックが同時に到達することを保証する領域（クロックドメイン）に分割する方法がとられているが，クロックドメイン間の通信に対してはそれぞれを同期する機構が必要である．そこで，コアやモジュールは従来の同期式設計を行い，コア間の通信にクロックを用いない非同期式設計を行う，大域非同期局所同期 (Globally Asynchronous, Locally Synchronous: GALS) 設計がその問題を解決する手段として知られている [1].

GALS 設計を用いて設計された GALS システムでは，従来のクロック分配やクロックスキューの問題が緩和され，高速でかつ多機能な SoC や NoC を効率的に

表 1.1 回路内部で非同期式通信を用いる割合に対する ITRS の予測 [1]

2009	2010	2011	2012	2013	2014	2015	2016
15%	17%	19%	20%	22%	23%	25%	30%
2017	2018	2019	2020	2021	2022	2023	2024
30%	30%	35%	40%	43%	45%	47%	49%

設計することが可能である．2009 年の ITRS(International Technology Roadmap for Semiconductors) の報告によると，2015 年には回路のうち 25% が，2024 年には 49% が非同期式の通信を用いてコア間のデータ転送が行われると予測されている (表 1.1)．本章では GALS システムを設計にするにあたり，VLSI のテストについて説明し，GALS システムに対するテスト品質の観点から GALS システムを構成する同期式回路，非同期式回路のテストにおける解決すべき問題をそれぞれ述べる．

1.2. VLSI のテスト

信頼性を向上させる 1 つの手段として，製品出荷前に行う VLSI のテストがある．VLSI のテストとは，回路中に故障が存在するか否かを判定することである．VLSI のテストは，回路に対して入力（テストパターン）を与え，その出力（出力応答）を観測することで行う．観測した出力応答と期待値を比較することで，回路中に故障が存在するか否かを判定することができる．期待値にはテストパターンを入力とする故障が存在しない回路でのシミュレーションにより得られる出力を用いる．また，テストパターンの質を評価する指標として，回路中に想定する故障と検出可能な故障の割合を示す故障検出率が用いられている．

回路の故障は様々な物理的要因により顕在化するため，それらを計算機で扱うことは困難である．そこで物理的要因を故障モデルとしてモデル化することを考える．故障モデルは一般に論理故障と遅延故障の 2 つに分類することができる．

論理故障は回路の論理に影響を与える故障であり，短絡や開放などの物理的な

欠陥をモデル化することが可能である．論理故障の代表的な故障モデルに，回路中の信号線が短絡や開放することによって 0 または 1 に固定される縮退故障がある．また，単一縮退故障モデルでは縮退故障が回路中にただ 1 つ存在するモデルを考えている．複数の縮退故障が同時に発生する同時故障においても，単一縮退故障を対象としたテストにより十分な数の故障を検出可能 [2] であることが知られている．単一縮退故障は古くから故障モデルの基本となっており，多くの研究がなされている．

遅延故障は回路の速度に影響を与える故障であり，信号線やゲートの遅延などの欠陥を検出することが可能である．近年，VLSI の高速化，プロセスルールの微細化に伴い，規定時間内に遷移が伝搬しない遅延故障が問題となってきている．遅延故障の故障モデルの 1 つにパス遅延故障がある．ここで，パスとは外部入力またはフリップフロップの出力から，外部出力またはフリップフロップの入力までの組合せ回路のみを通る経路であり，このパス上の微少な遅延が累積し，クロック周期を超える遅延がパス上に存在したときに活性化する故障をパス遅延故障と呼ぶ．パス遅延故障を検出することができれば，回路中の微少な遅延を検出することが可能である．

1.3. GALS システムのテスト

GALS 設計を用いて設計された SoC は，同期式コア，非同期式コア，それらを接続する非同期式ネットワークを用いて構成することができる．一般に SoC のテストでは各コアの入出力を制御するラッパーと，ラッパーに対してテストパターンの伝搬，出力応答の観測を実現するテストアクセス機構 (TAM) を用いることで，コアの外部入出力をチップの外部入出力として扱うことが可能である．図 1.1 に TAM とラッパーを挿入した GALS システムの一例を示す．

ラッパーと TAM を挿入することにより，各コア，非同期式ネットワークの外部入出力をチップの外部入出力として扱うことができるため，本論文では同期式回路部分，非同期式回路部分，それぞれを独立にテストすることを考える．つまり，GALS システムのテスト品質を向上させるためには，同期式回路部分，非同期式

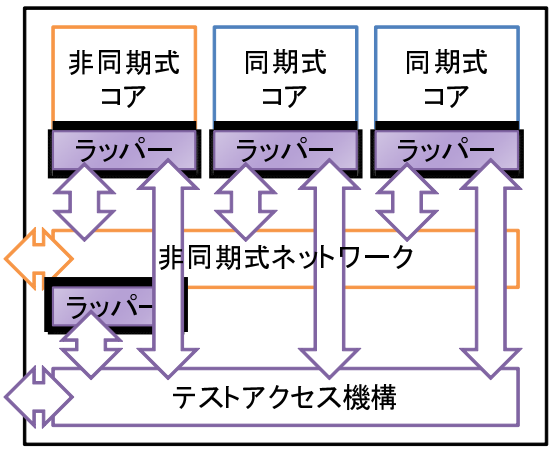

図 1.1　ラッパーと TAM を挿入した GALS システム

回路部分の両方に対して高いテスト品質を必要とする．本節では，同期式回路，非同期式回路に対する現状のテスト手法を紹介し，それぞれの問題点を述べる．

1.3.1　同期式回路部分のテスト

一般的な同期式順序回路に対してテストパターンを生成するアルゴリズムは複数提案されている．しかし，大規模な順序回路に対してそれらのアルゴズムを用いて，高い故障検出率を達成するテストパターンを生成することは困難である．組合せ回路に対して効率的にテストパターンを生成するアルゴリズムは提案されているため，順序回路をテスト時のみ組合せ回路として扱えるようにする完全スキャン設計が提案されている．完全スキャン設計を用いることで，回路は組合せ回路用のテスト生成アルゴリズムを用いることが可能となり，大規模な回路に対しても高速に，多くの故障を検出できる利点がある．

完全スキャン設計を行うことで，順序回路に対し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るが，組合せ回路用のテストパターンを完全スキャン設計された回路に印加すると，通常動作に影響のない故障が検出される場合がある．この故障が存在したとき，通常動作では正常に動作する回路が，テストでは不良と判定されるため，歩留まり損失が発生する．これはテスト品質の

観点から見ると，過剰テストを行っている．完全スキャン設計は非常に有用なテスト容易化設計手法であるが，過剰テストを発生させやすいという問題がある．

特に，パス遅延故障を考えたとき，順序回路には遷移が伝搬せず，活性化しないパス（フォールスパス）が多く含まれるため，完全スキャン設計を行い，組合せ回路としてパス遅延故障を検出するテスト生成を行うと，フォールスパスを検出するテストパターンが多く生成される．また，大規模回路中のパス数は膨大となり，パス遅延故障を検出するテストパターンを生成するテスト生成や，テスト実行には多くの時間が必要となる．そのため，順序回路中のフォールスパスを判定することができれば，フォールスパスをテスト対象から除くことで，過剰テストを緩和，テスト生成，テスト実行時間の削減を行うことが可能である．

1.3.2 非同期式回路部分のテスト

非同期式順序回路に対するテスト手法も複数提案されているが，同期式順序回路と同様に，大規模な順序回路に対するテストパターン生成は困難である．そのため，非同期式回路に対しても完全スキャン設計を適用することを考える．同期式回路で用いられる順序素子は，クロックに同期し値を取り込むフリップフロップが利用される．一方，非同期式回路にはDラッチ，C素子等，複数の種類の順序素子や，組合せループを用いて順序動作を実現している．

そのため，非同期式設計に対して組合せ回路用のテスト生成アルゴリズムを適用するためには，単に順序素子をスキャン素子に置き換えるだけでなく，非同期回路中に存在するすべての組合せループを切るように，スキャン素子を挿入する必要がある．文献 [3] では，レベルセンシティブスキャン設計 (Level Sensitive Scan Design: LSSD) で用いるスキャン素子を，各組合せループに挿入する手法が紹介されている．組合せループを切ることで，組合せ回路部分に対して組合せテスト生成手法を適用することができ，同期式回路に対するテスト手法と同等のテスト品質を保証することができるが，非常に多くのLSSDスキャン素子を挿入する必要があり，高い面積，遅延オーバーヘッドを招く問題がある．このオーバーヘッドを抑制するために，複数の完全スキャン設計手法が提案されているが，非同期式回路の組合せ回路部分の論理故障の検出が保証されない問題があり，テスト品

質が低いという問題がある．

本論文ではGALSシステムにおけるテスト品質を向上させるために，同期式回路に対する過剰テストの緩和と，非同期式回路に対するテスト不足の解消する手法を提案する．本論文の構成は以下の通りである．2章において，同期式回路に対する過剰テストの緩和を実現するために，高位で判定されたフォールスパスをゲートレベルで利用可能にするための手法を提案する．3章では，非同期式回路に対するテスト不足を解消する，完全スキャン設計手法を提案する．最後に，4章で本論文での結論を述べる．

第2章　RTLフォールスパスマッピング

2.1. 緒言

LSIの設計においてフォールスパスの情報を用いることで，設計やテストにおいて次の利点が得られる．設計の面では，フォールスパスについて設計制約を考慮する必要がないので，フォールスパス上に存在するゲートを遅延が大きく，面積が小さいゲートに変更することができ，回路面積を縮小できる場合がある．また，クリティカルパスより長いパスがフォールスパスと判定できた場合，そのパスに対しては設計制約を満たす必要がないため，このパスに関する論理最適化を省略できる．テストの面では，フォールスパス上のパス遅延故障に対するテスト生成はテストが存在しないために多大な時間を要するが，あらかじめフォールスパスと判定できればフォールスパスに対してテスト生成を行わずに済み，テスト生成時間を短縮することが可能である．また，テスト容易化設計に伴いフォールスパス上の遅延故障が検出可能になることがあるが，この遅延故障をテストから除外することで過剰テストを緩和することができる．

ゲートレベルでのフォールスパス判定法に，組合せ回路に対するフォールスパス判定法 [4, 5, 6] や，順序回路に対するフォールスパス判定法 [7, 8] などが提案されている．しかし，一般にゲートレベルでのフォールスパス判定手法は，大規模な回路に対して適用するのはパス数が膨大であるため困難である．そのため，ゲートレベルで膨大なパス数を扱うのではなく，ゲートレベルよりも要素数が少ないRTLの設計情報を用いてフォールスパスを扱う方法が提案されている [9, 10, 11, 12, 13]．文献 [9] はフォールスパスを特定する手法ではないが，回路の動作周波数を決めるタイミング解析において，RTLの設計情報を用いてクリ

ティカルパスより長いフォールスパスを，誤ってクリティカルパスと判定することを避ける方法を提案している．文献 [10] では，RTL でフォールスパスを定義し，RTL フォールスパスを判定する方法を提案している．さらに，Yu ら [11] はマルチサイクルパスを考慮した RTL フォールスパスを定義し，マルチサイクル RTL フォールスパスを判定する手法を提案している．また，高位合成情報を使った RTL フォールスパス判定法も提案されている [12, 13]．文献 [10, 11, 12, 13] の手法では，RTL のパスとゲートレベルのパスとの対応関係を保証するために，RTL 記述からゲートレベル回路を論理合成する際に，モジュールの境界を維持し，論理最適化をモジュール内でのみ行う，モジュール境界保存論理合成 [10] を用いている．現在のところ，RTL パスに対応するゲートレベルパスを探索する手法は提案されておらず，モジュール境界保存論理合成を用いることが，RTL パスに対応するゲートレベルパスを得る唯一の手法である．しかし，論理合成をモジュール境界保存論理合成に限定することは実用的でない．

本章では，モジュール境界保存論理合成を用いずに，RTL フォールスパスに対応するゲートレベルパスを探索する手法を提案する．RTL フォールスパスを任意の論理合成を用いて合成された回路に対し適用するために，まず，フォールスパスに限らず，RTL パスの集合に対応するゲートレベルパスの集合を探索する手法を提案する（パスのマッピングという）．提案手法では，RTL パスを構成する RTL 信号線と機能等価なゲートレベル信号線を探索する（信号線のマッピングという）．信号線の機能等価判定には多大な計算量を要するため，RTL パス集合の一意性を用いることや，機能等価判定を行う信号線の候補を絞り込むことで機能等価判定を行う回数を軽減する．RTL パス集合を一意に識別する RTL 信号線の数は RTL パス集合中に含まれる信号線数よりも少なく，限られた数の信号線のみをマッピングするため，信号線マッピングを行う回数を削減できる．信号線マッピングは，対象の信号線とゲートレベル回路中の全信号線との機能等価判定を行うことで実現できるが，これは実用的でない．そこで，故障診断技術を用いて機能等価信号線の候補を絞り込む手法 [14] を応用し，RTL 信号線と機能等価な信号線をゲートレベル回路中から効率的に探索する．

提案したパスマッピング手法を用いて得られたゲートレベルパスは，ゲートレ

ベル信号線の集合で表現されるため，各ゲートレベルパスを完全に記述する必要が無く，ゲートレベルパスを束として扱うことができる．この表現は Synopsys Design Constraint(SDC) 表現などで記述することができるため，EDA ツールと親和性が高い．実験結果では，多くの RTL パスが実用的な時間でゲートレベルパスにマッピングされることを示す．

次に，フォールスパスのマッピングを考える．文献 [10] で定義される RTL フォールスパスはモジュール境界保存論理合成を仮定しており，対応するゲートレベルパスがフォールスであることを保証している．本章では，提案するパスマッピング手法を用いて RTL フォールスパスをゲートレベルパスにマッピングした結果，得られたゲートレベルパスがフォールスであることを示す．実験結果では，任意の論理合成を用いて得られたゲートレベル回路中の，多くの RTL フォールスパスがゲートレベルフォールスパスにマッピングされたことを示す．

パスマッピング法を提案したことにより，RTL パスに対応するゲートレベルパスを得る手法は現在のところ 2 つ存在する．1 つはモジュール境界保存論理合成を行うことで，RTL フォールスパス情報をゲートレベル回路で完全に利用することが可能となるが，ゲートレベル回路を得る際に面積，遅延オーバーヘッドを必要とする．また，パスマッピング手法は，RTL パスに対応する，制約を与えない論理合成を用いて得られたゲートレベル回路中のゲートレベルパスを得ることができるため，面積，遅延オーバーヘッドは存在しないが，論理最適化の結果によっては RTL パスに対応するゲートレベルパスが存在しないことがある．そのため，RTL で得られたフォールスパスをゲートレベルで最大限利用するために，RTL フォールスパス情報を保存する新しい論理合成手法を提案する．

モジュール境界保存論理合成では，RTL 回路中のすべてのモジュール境界を保存していたが，提案する論理合成手法では与えられた RTL パス集合を一意に識別するために必要な最小個の RTL 信号線のみを保存する．実験結果では，提案する論理合成手法を用いて得られるゲートレベル回路は，完全なパスマッピングを保証しながら，面積，遅延オーバーヘッドの影響を抑制できたことを示す．

本章の構成は次のとおりである．まず，諸定義を 2.2 節で行う．2.3 節において，RTL パスマッピング手法を提案する．2.4 節において，提案したパスマッピング

手法を用いて，RTL フォールスパス集合からマッピングされたゲートレベルパスがフォールスであることを示し，2.5 節ではパスマッピング実験結果について述べる．2.6 節において，RTL フォールスパス情報を保存する論理合成手法を提案し，2.7 節において，提案した論理合成手法の評価実験を行う．2.8 節で本章の結言を述べる．

2.2. 諸定義

2.2.1 回路モデル

本章では構造記述された RTL 回路を対象とする．構造記述の RTL 回路は，コントローラとデータパスから成る．コントローラは組合せモジュールと状態レジスタから成り，データパスはレジスタ，演算モジュール（組合せ回路モジュール），マルチプレクサ，及びそれらの相互接続で構成される．コントローラ内の組合せモジュールと，データパス内の演算モジュール，マルチプレクサを単に組合せモジュールと呼ぶ．

2.2.2 ゲートレベルパスと RTL パス

定義 1 (ゲートレベルパス) 以下の条件を満たすゲートレベル信号線の順序集合 $\{e_1^G, \ldots, e_n^G\}$ を**ゲートレベルパス**と呼ぶ．

1. e_1^G は外部入力または FF の出力と隣接する信号線
2. e_n^G は外部出力または FF の入力と隣接する信号線
3. $e_i^G (2 \leq i \leq n-1)$ は e_{i-1}^G を入力として持つゲートと e_{i+1}^G を出力として持つゲート間を接続するゲートレベル信号線 □

定義 2 (部分ゲートレベルパス) ゲートレベルパス p^G の部分順序集合を $\boldsymbol{p^G}$ **の部分ゲートレベルパス**と呼ぶ． □

定義 3 (RTL パス) 以下の条件を満たす *RTL* 信号線の順序集合 $\{e_1^R, \ldots, e_n^R\}$ を **RTL パス**と呼ぶ.

1. e_1^R は外部入力またはレジスタの出力と隣接する *RTL* 信号線
2. e_n^R は外部出力またはレジスタの入力と隣接する *RTL* 信号線
3. $e_i^R (2 \leq i \leq n-1)$ は e_{i-1}^R を入力として持つモジュールと e_{i+1}^R を出力として持つモジュール間を接続する *RTL* 信号線 □

定義 4 (部分 RTL パス) *RTL* パス p^R の部分順序集合を **p^R の部分 RTL パス**と呼ぶ. □

ここで，RTL 信号線を構成する 1 ビットごとの RTL 信号線を以下のように定義する.

定義 5 (ビットスライス RTL 信号線) *RTL* 信号線 s を *1* ビットごとに分離した *RTL* 信号線を **s のビットスライス RTL 信号線**と呼び，s の i ビット目のビットスライス *RTL* 信号線を $s[i]$ で表す. □

定義 6 (ビットスライス RTL パス) 以下の条件を満たすビットスライス *RTL* 信号線の順序集合 $\{e_1^R[k_1], \ldots, e_n^R[k_n]\}$ を**ビットスライス RTL パス**と呼ぶ.

1. $e_1^R[k_1]$ は外部入力またはレジスタの出力と隣接する k_1 ビット目のビットスライス *RTL* 信号線
2. $e_n^R[k_n]$ は外部出力またはレジスタの入力と隣接する k_n ビット目のビットスライス *RTL* 信号線
3. $e_i^R[k_i] (2 \leq i \leq n-1)$ は $e_{i-1}^R[k_{i-1}]$ を入力として持つモジュールと $e_{i+1}^R[k_{i+1}]$ を出力として持つモジュール間を接続する k_i ビット目のビットスライス *RTL* 信号線 □

定義 7 (部分ビットスライス RTL パス) ビットスライス *RTL* パス p^R の部分順序集合を **p^R の部分ビットスライス RTL パス**と呼ぶ. □

2.2.3 信号線間の関係

機能等価信号線を定義するために必要な操作である信号線の切断を定義し，機能等価信号線を定義する．

定義 8 (信号線の切断) n 入力 m 出力の組合せ回路 C とその内部信号線 s について，次の操作を C の s での**切断**と呼ぶ．

1. C に対して新たに $n+1$ 番目の入力，$m+1$ 番目の出力をそれぞれ生成する
2. s を開放し，入力側の信号線を $m+1$ 番目の出力と接続し，出力側の信号線を $n+1$ 番目の入力と接続する □

この操作によって得られる組合せ回路を $C^*(s)$ で表す．

2つの機能等価な組合せ回路について，信号線の機能等価性を以下のように定義する．

定義 9 (機能等価信号線) 機能等価な組合せ回路 C_1，C_2 及びそれぞれの内部信号線 s_1, s_2 に対して，$C_1^*(s_1)$ と $C_2^*(s_2)$ が機能等価であるとき，かつそのときに限り $\boldsymbol{s_1}$ **と** $\boldsymbol{s_2}$ **は機能等価である**という． □

以降では，信号線 s_1 と s_2 の機能等価性を $s_1 \equiv_l s_2$ で表わす

図 2.1 は機能等価信号線を示している．信号線 s_1 と s_2 が機能等価であるためには，それぞれを切断した回路 $C_1^*(s_1)$ と $C_2^*(s_2)$ に対して，任意の入力パターンを印加したとき出力応答が常に等しくなる必要がある．

2.2.4 パス間の関係

部分ビットスライス RTL パスと部分ゲートレベルパスの機能等価関係を次のように定義する．

定義 10 (パスの機能等価関係) 部分ビットスライス *RTL* パス及び部分ゲートレベルパスを単に**部分パス**と呼ぶ．このとき，部分パス $q_1 = \{e_{1_1}, \ldots, e_{1_n}\}, q_2 = \{e_{2_1}, \ldots, e_{2_m}\}$ が以下の条件を満たすとき，$\boldsymbol{q_1}$ **と** $\boldsymbol{q_2}$ **は機能等価である**という．

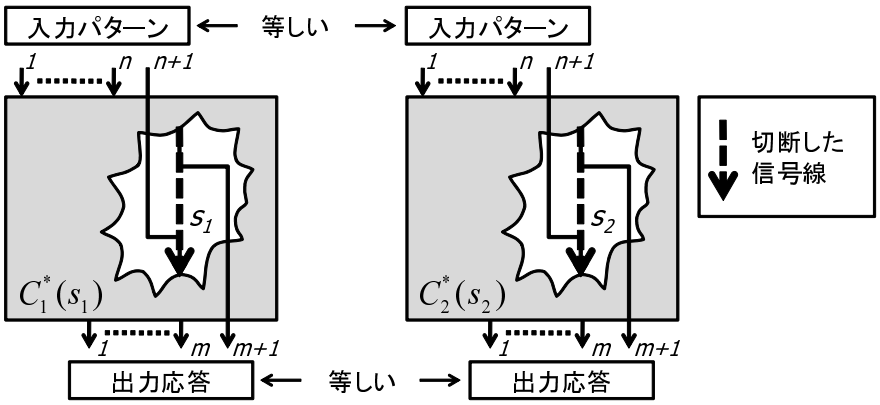

図 2.1 機能等価な信号線 s_1 と s_2

1. $n = m$

2. $e_{1_i} \equiv_l e_{2_i}$ $(i = 1, \ldots, n)$ □

RTL パスをゲートレベルパスへマッピングするためには，RTL パスを一意に識別する RTL 信号線をゲートレベル信号線にマッピングすれば十分である．信号線をマッピングする回数を削減するために，パスの識別を以下に定義する．

定義 11 (パスの識別) 部分 *RTL* パス q^R を部分集合とする *RTL* パス p^R がただ１つであるとき，$\boldsymbol{q^R}$ **は** $\boldsymbol{p^R}$ **を一意に識別する**という． □

定義 12 (パス集合の識別) 部分 *RTL* パス q^R を部分集合とする *RTL* パス集合 P^R がただ１つであるとき，$\boldsymbol{q^R}$ **は** $\boldsymbol{P^R}$ **を一意に識別する**という． □

図 2.2 に，RTL データパスの組合せ回路部分の 10 本のパスと信号線を用い，それらの関係を例示する．RTL パス $p^R = \{a, d, f, g\}$ を考えたとき，p^R の部分 RTL パスの例としては $\{a, d, g\}$ がある．RTL 信号線 d はビットスライス RTL 信号線 $d[0]$ と $d[1]$ から成り，ビットスライス RTL パス $\{a[1], d[0], f[1], g[1]\}$ の部分ビットスライス RTL パスの例としては $\{d[0], f[1]\}$ がある．部分ビットスライス RTL パス $q_1 = \{d[0], f[1]\}$ と部分ゲートレベルパス $q_2 = \{D[0], F[1]\}$ を考えたとき，$d[0] \equiv_l D[0]$ かつ $f[1] \equiv_l F[1]$ であれば，q_1 と q_2 は機能等価である．部分 RTL パ

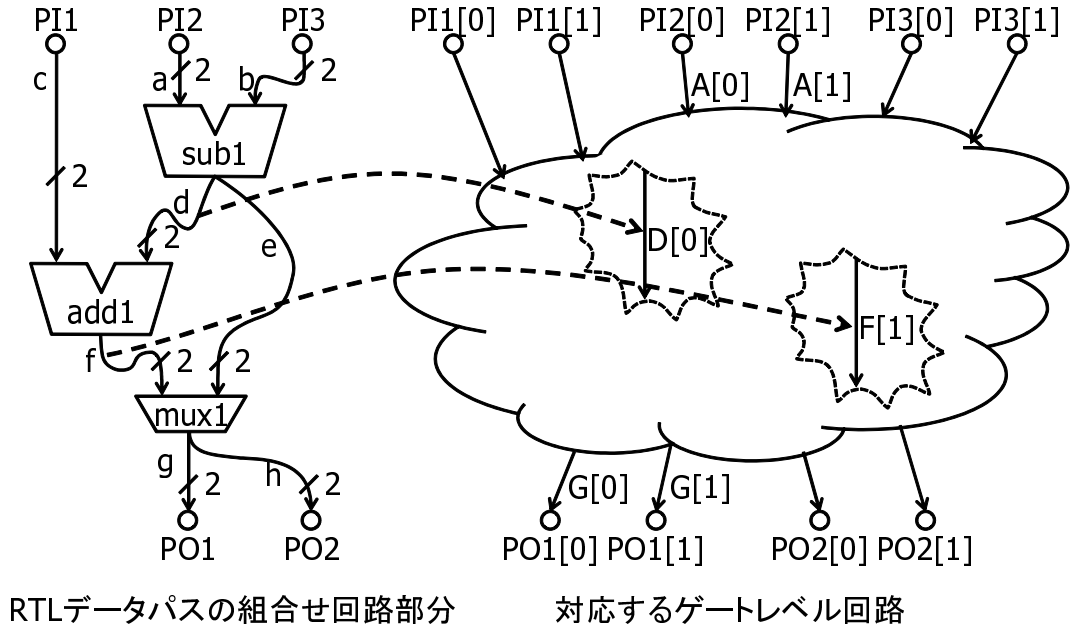

図 2.2 RTL データパスとそれに対応するゲートレベル回路の例

ス $\{a, d, g\}$，$\{a, f, g\}$ は $p^R = \{a, d, f, g\}$ を一意に識別する．同様に，部分 RTL パス $\{a, g\}$ は，RTL パス集合 $P^R = \{\{a, d, f, g\}, \{a, e, g\}\}$ を一意に識別する．

2.3. 提案するパスマッピング法

本節ではパスマッピング問題を定式化し，パスマッピングを実現する手法をフォールスパスとは独立に提案する．フォールスパスに対する本提案法の応用は 2.4 節で述べる．

2.3.1 パスマッピング問題

RTL パス集合に対応するゲートレベルパスを探索する手法を，パスマッピング問題として定式化する．

マッピングに用いるRTL回路は，与えられた構造記述RTL回路S^Rから組合せ回路部分を取り出したRTL組合せ回路C^Rとする．また，ゲートレベル回路は，S^Rを論理合成したゲートレベル回路S^Gから組合せ回路部分を取り出したゲートレベル組合せ回路C^Gを扱う．ここで，C^Rの外部入出力の各ビットスライスRTL信号線とC^Gの外部入出力の間に全単射の関係があるものとし，この関係を**入出力マッピング情報**と呼ぶ．C^RとC^Gの入出力マッピング情報は，S^RからS^Gを論理合成する際に，S^Rのレジスタのすべてのビットと，S^GのすべてのFFの間に全単射の関係を保証することで得ることができる．この入出力マッピング情報が得られない場合，提案するパスマッピング法を適用することはできないが，構造記述のRTL回路に対して論理合成を行う場合，一般にこの関係は保証される．

定義 13 (パスマッピング問題)

入力
- C^R: *RTL*組合せ回路
- C^G: C^Rと機能等価なゲートレベル組合せ回路
- C^RとC^Gの入出力マッピング情報
- P^R: *RTL*パス集合

出力 $P^G = \bigcup_{i=0}^{n} \bigcup_{j=0}^{m_i} P_{ij}^G$, ただし，$P_{ij}^G$は以下のように定義される．$q_i^R(i = 1, \ldots, n)$を$P^R$を一意に識別する部分*RTL*パスとし，$q_{ij}^R(j = 1, \ldots, m_i)$を$q_i^R$の部分ビットスライス*RTL*パスとする．$n$は$P^R$の部分*RTL*パスの数，$m_i$は$q_i^R$中のすべての*RTL*信号線のビット位置を指定したときに得られるビットスライス*RTL*パスの組合せである．q_{ij}^Gはq_{ij}^Rと機能等価な部分ゲートレベルパスであり，P_{ij}^Gはq_{ij}^Gを含むゲートレベルパスの集合である． □

2.3.2 パスマッピングアルゴリズム

RTLパス集合P^Rに対応するゲートレベルパス集合P^Gを探索するパスマッピング問題を解くアルゴリズムを次に示す.

1. P^Rを一意に識別する最小の部分RTLパス$q_i^R(i = 1, \ldots, n)$を列挙する

2. q_i^R のビットスライス RTL パス $q_{ij}^R(j = 1, \ldots, m_i)$ 中の各ビットスライス RTL 信号線 $e_{ij_k}^R(k = 1, \ldots, l)$ と機能等価なゲートレベル信号線 $e_{ij_k}^G$ を得る．ただし，m_i は q_i^R 中の各 RTL 信号線のビット位置を指定して得られる組合せの数であり，l は q_i^R 中の RTL 信号線の数である．少なくとも 1 つの部分 RTL パス q_i^R 中の，すべての j と k に対して $e_{ij_k}^G$ が得られた場合，つまり，部分 RTL パス中のすべてのビットスライス RTL 信号線がゲートレベル信号線にマップされた場合，Step 3 を行う．(1 つも見つからない場合は，すべての q_i^R について Step 2 を行う)

3. Step 2 において，対応する部分ビットスライス RTL パス $\left\{e_{ij_1}^R, \ldots, e_{ij_l}^R\right\}$ の $e_{ij_k}^R$ からマップされたすべての $e_{ij_k}^G\ (k = 1, \ldots, l)$ である各部分ゲートレベルパス $\left\{e_{ij_1}^G, \ldots, e_{ij_l}^G\right\}$ について，$\left\{e_{ij_1}^G, \ldots, e_{ij_l}^G\right\}$ によって一意に識別されるすべてのゲートレベルパスを探索する．このゲートレベルパス集合を P_{ij}^G とする．

4. $P^G = \bigcup_{i=0}^{n} \bigcup_{j=0}^{m_i} P_{ij}^G$ を得る．

P^R を一意に識別する最小の部分 RTL パス $q_i^R(i = 1, \ldots, n)$ を得る手法を以下に示す．サイズ s の長さの部分 RTL パスが P^R を一意に識別するかどうかを判定し，サイズ s の長さで一意に識別できた場合，その部分 RTL パスをすべて列挙する．サイズ s で見つからない場合は，s をインクリメントして判定を続ける．ただし，s の初期値は 0 である．

RTL 回路中の RTL モジュール数はゲートレベル回路中のモジュールに比べ非常に少ない．そのため，最小の RTL パス集合を得るために必要な時間は少ないと考えられ，実験結果において処理の要した時間を評価する．Step 2 で行う信号線のマッピング手法は次の副節で提案する．ここで，アルゴリズムの記述を簡単にするために，ビットスライス RTL 信号線と機能等価なゲートレベル信号線は多くとも 1 つであると仮定する．2.5 節で示す実験結果からは，機能等価なゲートレベル信号線が 2 つ以上存在することはなかったが，複数の信号線が得られた場合においても，それらの信号を通るすべてのパスを扱うことで対応することができる．Step 3 と 4 では，ゲートレベルパスを列挙すると述べているが，大規模

ゲートレベル回路においてゲートレベルパスを列挙することは現実的ではない．提案するアルゴリズムでは，通過するゲートレベル信号線を指定することでゲートレベルパスを表現する．パスは通過する一部の信号線を $\left\{e^G_{ij_1}, \ldots, e^G_{ij_l}\right\}$ のように指定することで指定することができる．この表現はSDC表現などのEDAツールと親和性が高い．

図 2.2 に示す RTL 回路とゲートレベル回路を用いて，RTL パス集合 $P^R = \{\{a, d, f, g\}\}$ に対するパスマッピングの例を示す．Step 1 において，P^R を一意に識別する最小の部分 RTL パス $q^R_1 = \{a, d, g\}$，$q^R_2 = \{a, f, g\}$ を得る．Step 2 では，まず，q^R_1 上のビットスライス RTL 信号線 $a[0], a[1], d[0], d[1], g[0], g[1]$ と機能等価なゲートレベル信号線を探索する．ここで，$a[0], a[1], d[0], d[1], g[0], g[1]$ と機能等価なゲートレベル信号線 $A[0], A[1], D[0], G[0], G[1]$ がそれぞれ見つかったとし，$d[1]$ と機能等価なゲートレベル信号線は見つからなかったとする．q^R_1 上のすべての RTL 信号線をマップすることができなかったので，もう一方の部分 RTL パス q^R_2 を用いて Step 2 を繰り返す．つまり，$f[0]$ と $f[1]$ に対して信号線マッピングを行う．信号線マッピングの結果，$f[1]$ と機能等価なゲートレベル信号線 $F[1]$ が見つかり，$f[0]$ と機能等価なゲートレベル信号線は見つからなかったとする．P^R を一意に識別する最小の部分 RTL パスをすべて試行したので，Step 3 および Step 4 を行う．このとき，P^R に対応するゲートレベルパス集合は，通過するゲートレベル信号線を以下のように指定することで得られる．$\{A[0], D[1], G[0]\}$, $\{A[0], D[1], G[1]\}$, $\{A[1], D[1], G[0]\}$, $\{A[1], D[1], G[1]\}$, $\{A[0], F[0], G[0]\}$, $\{A[0], F[0], G[1]\}$, $\{A[1], F[0], G[0]\}$, $\{A[1], F[0], G[1]\}$

2.3.3 信号線マッピング

本副節では，機能等価な信号線を探索する問題を信号線マッピング問題として定式化し，信号線マッピング問題を解くアルゴリズムを提案する．信号線マッピングアルゴリズムは提案したパスマッピングアルゴリズムで用いる．

信号線マッピング問題

RTL 回路中のビットスライス RTL 信号線と機能等価なゲートレベル信号線をゲートレベル回路から探索する問題を**信号線マッピング問題**として以下のように定式化する．

定義 14 (信号線マッピング問題)

入力
- C^R: *RTL* 組合せ回路
- C^G: C^R と機能等価なゲートレベル組合せ回路
- C^R と C^G の入出力マッピング情報
- $e^R[k]$: *RTL* 信号線 e^R の k ビット目のビットスライス *RTL* 信号線

出力 $E^G = \left\{e^G | e^G \equiv_l e^R[k]\right\}$ ただし，e^G は C^G 中のゲートレベル信号線 □

信号線マッピングアルゴリズム

RTL 組合せ回路 C^R とゲートレベル組合せ回路 C^G が与えられたとき，C^R 中のビットスライス RTL 信号線 $e^R[k]$ と C^G 中のゲートレベル信号線 e^G との機能等価判定は，$C^{R*}(e^R[k])$ と $C^{G*}(e^G)$ に対して任意の入力パターンを印加したときの出力応答を比較することで行うことができる．$C^{R*}(e^R[k])$ と $C^{G*}(e^G)$ に対する機能等価判定は文献 [15, 16] などの実用的な検証技術を用いることができるが，$e^R[k]$ と C^G 中の e^G の可能な組合せすべてについて，機能等価判定を行うことは現実的ではない．

文献 [14] では，故障診断技術を用いた，ビットスライス RTL 信号線と機能等価なゲートレベル信号線の候補をゲートレベル回路中から探索する手法を提案している．この手法は，ビットスライス RTL 信号線とゲートレベル信号線のすべての対について，すべての入力パターンの組合せを用いた機能等価判定をするのではなく，テストパターン集合 T と縮退故障の機能等価性を用いた，限られた入力パターンに対して機能等価な信号線の候補を探索する．本論文ではこの手法を応用し，信号線マッピング手法を解く手法を提案する．文献 [14] では，ビットス

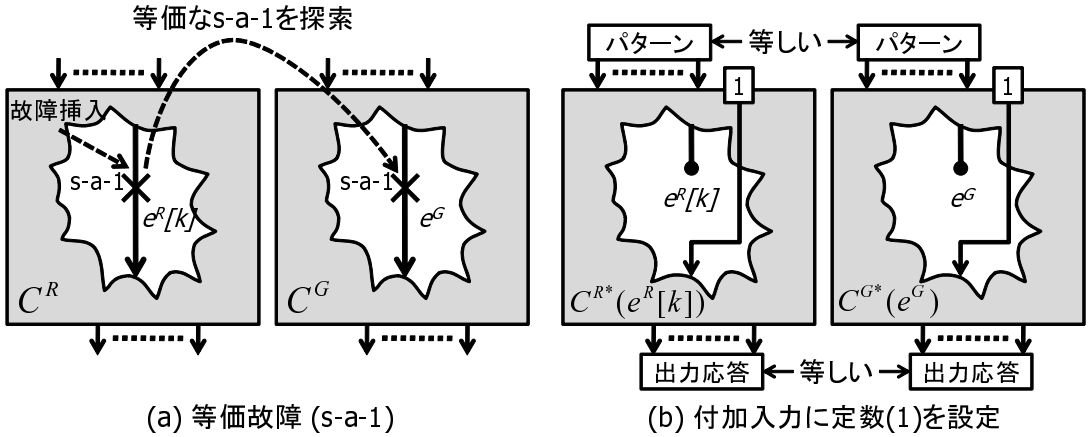

図 2.3 信号線の機能等価性の必要条件としての故障の等価性

ライス RTL 信号線 $e^R[k]$ に対して縮退故障 f^R を仮定し，ゲートレベル回路中からその故障と T に関して等価な縮退故障 f^G を探索することを考える[1]．すなわち，$e^R[k]$ 上に挿入した s-a-v 故障と T に関して等価なゲートレベル信号線 e^G 上の s-a-v 故障を見つけることができれば (図 2.3(a) 参照)，T に関して機能等価な信号線であるための必要条件である，$e^R[k]$ と e^G の値を v に固定し，$t \in T$ を印加したときの RTL 回路とゲートレベル回路の出力応答が等しくなる (図 2.3(b) 参照) ことが成り立つ.

以下に信号線マッピング問題を解く全体のアルゴリズムを示す.

1. C^G 中のすべての検出可能な単一縮退故障を検出するテスト集合 T を求める
2. 各 $v \in \{0, 1\}$ に対して，次の 2 つの手順を適用する
   (a) $e^R[k]$ に縮退故障 s-a-v を挿入し，T を入力とする RTL シミュレーションを行い，出力応答の集合 R_{fv} を求める
   (b) T を印加すると R_{fv} を出力する C^G 内の s-a-v 故障の存在する信号線集合 E^{Gv} を求める

[1]機能等価な RTL 回路 C^R とゲートレベル回路 C^G に対して，それぞれ f^R, f^G を挿入した回路が機能等価になるような故障 f^R, f^G を**等価故障**と呼ぶ.

3. $E^G = E^{G0} \cap E^{G1}$ を求める

4. 各 $e^G \in E^G$ について，C^R 中の $e^R[k]$ と，C^G 中の e^G をそれぞれ切断し，$C^{R*}(e^R[k])$ と $C^{G*}(e^G)$ を作成する

5. $C^R(e^{R*}[k])$ と $C^{G*}(e^G)$ に対して機能等価判定を行い，機能等価でない信号線 e^G を E^G から取り除く

Step 1 から 3 の手続きは，文献 [14] で提案されている故障診断を用いた手続きと同等である．文献 [14] では，故障診断に用いる入力パターンとして，C^G 中のすべての検出可能な単一縮退故障を検出する完全テスト集合 T を用いている．まず，C^R 中の $e^R[k]$ に対して s-a-0(1) 故障を挿入し，C^G 中から挿入した故障と等価な s-a-0(1) 故障を探索する．この手続きで得られるゲートレベル信号線は，$e^R[k]$ と機能等価であるための必要条件を満たす．最後に，Step 4 と 5 の手続きにおいて，前述の手続きにおいて絞られた候補に対して機能等価判定を行う．

アルゴリズムの完全性を定理 1 に示す．信号線マッピングアルゴリズムの Step 1 において，完全テスト集合を用いている．テスト集合の不完全性は信号線マッピングの正当性に影響を与えないが，完全テスト集合が得られない場合，機能等価判定（Step 4 と 5）の試行回数が増加するため，故障診断技術を用いて効果的に機能等価な信号線の候補を絞り込む（Step 2b と 3）ためには完全テスト集合が得られる方が望ましい．ここで，アルゴリズムで用いる故障診断技術を用いると，対象の故障をすべて得られるものとする．つまり，故障診断ツールは与えられた入力パターンに関して等価故障を見逃さないものとする．不完全な対象の故障を見逃す故障診断ツールを用いる場合，提案する信号線マッピングアルゴルズムでは，ゲートレベル回路中に存在する機能等価信号を得られない．パスマッピングの観点から考えると，対応するゲートレベルパスと判定可能なパス数の削減につながる．故障診断ツールが等価でない故障を出力する場合，すべての機能等価信号線の候補について機能等価判定を行うため，信号線マッピングの正当性に影響を与えない．

定理 1 *RTL* 組合せ回路 C^R とその内部信号線 $e^R[k]$ と，C^R と機能等価なゲートレベル組合せ回路 C^G に対して，信号線マッピングアルゴリズムによって得られ

る C^G中の 各ゲートレベル信号線 $e^G \in E^G$ は，$e^R[k]$ と機能等価であり，かつこの集合に限る．□

[**証明**]

まず，Step 1 から 3 の手続きにおいて，e^G と $e^R[k]$ が同じ値を持ち，T に含まれる任意の入力パターンを C^G と C^R の外部入力印加したとき，C^G と C^R の外部出力における出力応答が等しいことを示す．C^R を n 入力 m 出力とし，入力を $x^R[i](i = 1, \ldots, n)$，出力を $z^R[i](i = 1, \ldots, m)$ とする．また，$C^{R*}(e^R[k])$ の入力を $x^{R*}[i](i = 1, \ldots, n+1)$，出力を $z^{R*}[i](i = 1, \ldots, m+1)$ とする．C^R の信号線 $e^R[k]$ に対して s-a-v(ただし，$v \in \{0, 1\}$) を挿入した場合を考える．任意の $t \in T$ を s-a-v を挿入した C^R の入力に印加して得られる出力 $z^R[1], \ldots, z^R[m]$ の出力応答と，$C^{R*}(e^R[k])$ の入力に $t\&v$ を印加して得られる $z^{R*}[1], \ldots, z^{R*}[m]$ の出力応答は等しい．ただし，“$a\&b$” の操作は a と b のビット結合操作を示す．

同様に，n 入力 m 出力の C^G の入力を $x^G[i](i = 1, \ldots, n)$，出力を $z^G[i](i = 1, \ldots, m)$ とし，$C^{G*}(e^G)$ の入力を $x^{G*}[i](i = 1, \ldots, n+1)$，出力を $z^{G*}[i](i = 1, \ldots, m+1)$ とする．ここで e^G に s-a-v が存在するとき，任意の $t \in T$ を s-a-v を挿入した C^G の入力に印加して得られる出力 $z^G[1], \ldots, z^G[m]$ の出力応答と，$C^{G*}(e^G)$ の入力に $t\&v$ を印加して得られる $z^{G*}[1], \ldots, z^{G*}[m]$ の出力応答は等しい．よって $e^R[k]$ 上の s-a-v と e^G 上の s-a-v が T に関して等価な故障であれば，C^R と C^G が機能等価であることから，$t^* \in T^*$ を印加したときの $C^{R*}(e^R[k])$ と $C^{G*}(e^G)$ の外部出力 $z^{R*}[1], \ldots, z^{R*}[m]$ と $z^{G*}[1], \ldots, z^{G*}[m]$ の出力応答は常に等しい．

続いて，C^R と C^G は機能等価であり，T の下で $e^R[k]$ 上の s-a-v 故障と e^G 上の s-a-v 故障は等価であるため，T の任意のパターンに対して，$C^{R*}(e^R[k])$ の外部出力 $z^{R*}[1], \ldots, z^{R*}[m]$ と，$C^{G*}(e^G)$ の外部出力 $z^{G*}[1], \ldots, z^{G*}[m]$ それぞれにおける出力応答は常に等しい．これは，$e^R[k]$ と e^G が機能等価であるための必要条件である．

故障診断での仮定より，上記の条件を満たすすべての e^G は E^G として得ることができるため，E^G は明らかに，$e^R[k]$ と機能等価なすべてのゲートレベル信号線を含む．そのため，Step 4 と 5 の手続きが，$e^R[k]$ と機能等価でないゲートレベル信号線を取り除き，かつその集合に限ることを示せばよい．明らかに，e^G が

$e^R[k]$ と機能等価でなければ E^G から取り除かれ，そうでなければ取り除かれない．よって定理は証明された． □

2.4. RTL フォールスパスマッピング

本節では，提案したパスマッピングアルゴリズムの応用として，RTL で判定されたノンロバストテスト不能パスのゲートレベルへのマッピングを示す．テストの観点からは，市販の ATPG ツールは機能的活性化不能の条件を満たすパス遅延故障を検出するテストパターンを検出できないため，ノンロバストテスト不能パスを判定することは重要である．本節ではノンロバストテスト不能パスをフォールスパスとして扱う．文献 [17] において，吉川らは RTL 回路におけるノンロバストテスト不能パスを以下のように定義している．

定義 15 (RTL ノンロバスト不能パス) *RTL* 回路 S^R から論理合成して得られるゲートレベル回路 S^G 中の $\delta(p)$ に含まれるゲートレベルパスがすべてノンロバストテスト不能であるとき，S^R 中の *RTL* パス p は *RTL* ノンロバストテスト不能である．ただし，$\delta(p)$ は p に対応するゲートレベルパスの集合とする． □

吉川らは，RTL ノンロバストテスト不能パスと $\delta(p)$ の関係を保証するために，モジュール境界保存論理合成を用いている [10]．論理合成に仮定を置いた状態で，RTL ノンロバストテスト不能パスであるための十分条件をマルチプレクサとレジスタの制御信号線に基き，以下のように提案している．

RTL 回路中のパス $p = \{e_1, \ldots, e_n\}$ が与えられたとき，任意の入力系列と任意の時間 t において，以下の４つの条件のうち少なくとも１つを満たせば，p は RTL ノンロバストテスト不能パスである．

1. t と $t+1$ の間に，e_1 を駆動する始点レジスタに遷移を起こす可制御性が存在しない
2. $t+1$ において，任意の i $(i = 1 \ldots n)$ について e_{i+1} の値が e_i の値と独立である

3. $t+2$ において，e_n に現れる値が終点レジスタによって取り込まれない

4. $t+2$ において，e_n に隣接するレジスタに取り込まれた e_n の値が，任意の外部出力に影響を与えない

これらの条件はコントローラから与えられるマルチプレクサ，レジスタの制御信号線を調べることによって確認できる．ここで，対象とする RTL 回路モデルは，コントローラの状態遷移が既知であり，状態遷移は状態と入力ベクタの対によって遷移先を指定するモデルを考えている．文献 [17] では，RTL ノンロバストテスト不能パスの詳細な条件や，対象とする回路モデルがより詳しく説明されている．

この条件を満たす RTL パスは，ノンロバスト活性化条件を満たすいかなる遷移が伝搬しない，もしくは，終点レジスタによって取り込まれた応答が外部出力において観測されないことを示している．提案したパスマッピング手法を用いることによって論理合成の仮定を取り除き，文献 [10] で提案されている RTL ノンロバスト不能パス判定手法を，さらに多くの一般的な回路に適用できる．

定理 2 *RTL* 回路 S^R 中の *RTL* ノンロバストテスト不能パス p^R について，提案したパスマッピング法を用いてマッピングされた p^R に対応するゲートレベルパス $p^G \in P^G$ はノンロバストテスト不能パスである． □

証明 p^R と p^G をそれぞれ，$\{e_1^R, \ldots, e_n^R\}$，$\{e_1^G, \ldots, e_m^G\}$ と表現し，p^R の各 RTL 信号線 e_i^R は任意のビット幅を持つとする．RTL ノンロバストテスト不能パスの十分条件から，任意の入力系列について，p^R は上述の 4 つの条件のうち少なくとも一つを満たす．I/O マッピング情報が存在するという仮定から，S^R と S^G の組合せ回路部分は機能等価である．

p^R が条件 1. を満たす場合，e_1^R のすべてのビットスライス RTL 信号線は時間 t から $t+1$ にかけて遷移を発生することができない．よって，e_1^G は t から $t+1$ にかけていかなる遷移を発生することができない．p^R が条件 3. もしくは条件 4. を満たす場合，$t+1$ における e_n^R のすべてのビットスライス RTL 信号線の値は外部出力によって観測できない．p^R が条件 3. を満たす場合，I/O マッピング情報

と S^R と S^G の組合せ回路部分の機能等価性，および信号線の機能等価性により，e_n^R のすべてのビットスライス RTL 信号線は，e_1^R から始まり（e_i^R を通過し）e_n^R に到達する遷移を持つことができない．ただし，e_i^R はパスマッピングを行うにあたり，信号線マッピングが必要となる信号線である．よって，e_m^G は t から $t+1$ において，e_1^G から始まり（e_j^G を通過し）e_m^G に到達する遷移を持つことができない．ただし，e_j^G は e_i^R からマッピングされるゲートレベル信号線である．よって，定理は証明された． □

この定理により，提案したパスマッピング手法と RTL ノンロバステスト不能パスを利用し，任意の論理合成を用いて得られたゲートレベル回路中のノンロバステスト不能パスを扱うことができる．

2.5. RTL パスマッピングと RTL フォールスパスマッピングの評価実験

本節では提案した RTL パスマッピング法を評価するために，RTL パスのマッピングと，文献 [10] で提案されている RTL フォールスパス判定手法を用いて判定された RTL フォールスパスのマッピングを行う．評価に用いる回路としては，3 つの RTL ベンチマーク回路 (LWF, Tseng, Paulin) と 1 つの実回路 (MPEG) のデータパス部分を用いる．表 2.1 にそれぞれの回路特性を示す．ただし，“#bit” はビット幅，“#PI” は外部入力数，“#PO” は外部出力数，“#reg” はレジスタ数を示す．“回路 (#gates)” の下の “MIP-LS” はモジュール境界保存論理合成を用いて論理合成した際の面積，“MIB-LS” は制約なしで論理合成した際の面積を示す．提案法により，論理合成に制約を与えずにパスマッピングが行えるので，面積を削減できることが示された．

実験では論理合成ツールとして Synopsys 社の Design compiler Y-2006.06-SP4 を用い，制約を指定しない論理合成を行った．ATPG ツールとして Synopsys 社の TetraMax を用い，論理合成後のゲートレベル回路中のすべての検出可能な単一縮退故障を検出するテストパターンを生成した．故障診断ツールとしては，Cadence 社の Encounter Test & Diagnostics を用いた．等価検証ツールとしては，Synopsys

表 2.1 各回路のデータパス部の特性

回路	#bit	#PI	#PO	#reg	面積 (#gate)	
					MIP-LS	MIB-LS
LWF	16	2	2	5	1,571	1,467
Tseng	8	3	2	6	1,357	1,077
Paulin	8	2	2	7	1,590	1,303
MPEG	8	5	16	241	38,183	28,454

表 2.2 パスマッピング結果

	LWF	Tseng	Paulin	MPEG
Pmr[%]	73.7	90.0	100.0	100.0
Pmr_b[%]	74.2	96.8	100.0	100.0
CPU[sec]	28.14	21.74	0.30	0.10

社の Formality を用いた．また，ゲートレベル上のパスを列挙するために Synopsys 社の PrimeTime を用いた．使用した計算機は，すべて Sun Microsystems 社の SunFire X4100 (Dual Opteron 256, 3.0GHz, 16GB) を用いた．

パスマッピングを行うにあたり，評価基準としてパスマッピング率 $Pmp = \frac{|P^{RM}|}{|P^R|} \times 100[\%]$ を用いる．ただし，$|P^R|$ をデータパス中の RTL パス総数，$|P^{RM}|$ をマッピングされた RTL パス数とする．更に詳細に評価するために，ビットスライス RTL パス単位でもパスマッピング率を考える．その評価基準として，ビットスライスパスマッピング率 $Pmp_b = \frac{|P_b^{RM}|}{|P_b^R|}$ を用いる．ただし，$|P_b^R|$ をデータパス中のビットスライス RTL パス総数，$|P_b^{RM}|$ をパスマッピングされたビットスライス RTL パス数とする．表 2.2 に 4 つの RTL ベンチマーク回路のデータパスに対して提案手法を適用し，計算したパスマッピング率とそれに要した時間を示す．

表 2.3 に信号線マッピングとパスマッピング結果の詳細を示す．ただし，“#Ptotal” は RTL パス総数，“#Punique” は I/O マッピング情報を用いて一意に識別されるパス数，“#Stried” は信号線マッピングを行う必要のある信号線数，

表 2.3 信号線マッピングとパスマッピングの詳細

	LWF		Tseng		Paulin		MPEG	
	RTL	bsRTL	RTL	bsRTL	RTL	bsRTL	RTL	bsRTL
#Ptotal	19	4,600,384	20	36,448	29	123,600	606	326,176
#Punique	14	3,412,544	18	31,840	29	123,600	606	326,176
#Stried	5	80	5	40	0	0	0	0
#Smapped	0	13	0	12	-	-	-	-
#Pmapped	14	3,415,360	18	35,296	29	123,600	606	326,176

“#Smapped” はマッピングされた（機能等価な信号線が見つかった）信号線数，“#Pmapped” はマップされたパス数を示す．回路名の下の “RTL” と “bsRTL” はビット幅を持つ束の RTL とビットスライス RTL を示す．RTL パスに含まれるビットスライス RTL パスの数は，RTL パスを構成する信号線のビット幅の直積を取ることによって算出した．定義 11（パスの一意性）により，多くの RTL パスが I/O マッピング情報のみを用いてマッピングすることができ，計算時間を短縮することができた．LWF の 2 本のパスと Tseng の 2 本のパスをマッピングするために，信号線マッピングをそれぞれ 80 回と 40 回行った．信号線マッピングに要した時間の平均は LWF が 0.35 秒であり，Tseng が 0.54 秒であったため，それぞれの回路に対する総計算時間がほぼ等しくなった．提案手法は平均，RTL パスマッピング率 90.9% ，ビットスライス RTL パスマッピング率 91.0% を達成した．ここで，マッピングされなかったパスについて考察する．ゲートレベルパスにマップされなかった（ビットスライス）RTL パスが存在する理由は，アルゴリズムがパスマッピングに必要な RTL 信号線信号線と機能等価なゲートレベル信号線をゲートレベル回路中から発見できなかったことに起因する．つまり，パスマッピングに必要なビットスライス RTL 信号線と機能等価な信号線がゲートレベル回路中に存在しなかったためである．

表 2.4 にフォールスパスマッピングの結果とマッピングに要した時間を示す．ただし，“#Ptotal” は RTL パス総数，“#Pfalse” フォールスパス数，“Ratio” は総 RTL パスに対するフォールスパスの割合，“Total” はフォールスパスマッピングに要した合計時間，“Ravi” は機能等価信号線の候補を探索するために要した

表 2.4 フォールスパスマッピング結果

	LWF		Tseng		Paulin		MPEG	
	RTL	Gate level	RTL	Gate level	RTL	Gate level	RTL	Gate level
#Ptotal	19	1,845,916	20	856,116	29	2,307,064	606	1,784,824
#Pfalse	5	470,300	6	418,752	13	1,610,968	32	16
Ratio[%]	26.32	25.48	30.00	48.91	44.83	69.83	5.28	0.00
Total[s]	15.36		21.73		0.27		1.72	
Unique[s]	0.21		0.24		0.27		1.72	
Ravi[s]	15.15		17.07		0.00		0.00	
FEchk[s]	0.00		4.42		0.00		0.00	
Pwhole[s]	93.21		37.21		103.39		303.65	
Pfalse[s]	24.26		19.07		73.53		0.22	

時間，“FEchk” は機能等価判定に要した時間，“Pwhole” は総パスを数え上げるために要した時間，“Pfalse” はマッピングされたフォールスパス数を数え上げるために要した時間を示す．回路名の下の “RTL” と “Gate level” はそれぞれ RTL でのパス数とゲートレベルパスでのパス数を示す．実験結果により，提案したパスマッピング法を用いることで，実用的な時間で，モジュール境界保存論理合成を仮定せずに多くのゲートレベルフォールスパスを得ることができた．

一方で順序テスト生成アルゴリズムもゲートレベルフォールスパスを判定することができるが，順序テスト生成ツールは実用的な時間では多くのゲートレベルフォールスパスを得ることができない．文献 [18] に挙げられる例では，TetraMax は Paulin の 10,000 フォールスパスを判定するために 50 時間を要している．文献 [10] で提案されている RTL フォールスパス判定手法と提案したパスマッピング手法を用いた場合，複数の回路で 1 秒以下での判定が可能であり，高位でのフォールスパス判定手法は非常に有効であることを示している．

表 2.5 にフォールスパスマッピングの詳細を示す．ただし，“#Pfalse” は総 RTL フォールスパス数，“#Punique” は I/O マッピング情報のみを用いてマッピングされたパス数，“#Stried” は信号線マッピングを試行したビットスライス RTL 信号線数，“#Smapped” はマッピングできた RTL 信号線数を示す．よって，提案

表 2.5 フォールスパスマッピングの詳細

	LWF	Tseng	Paulin	MPEG
#Pfalse	5	6	13	32
#Punique	4	5	13	32
#Stried	32	16	0	0
#Smapped	0	7	-	-

表 2.6 論理合成の特徴

論理合成タイプ	マッピング率	面積オーバーヘッド	遅延オーバーヘッド
MPP-LS	完全	低い	低い
MIP-LS	完全	高い	高い
MIB-LS	不完全	なし	なし

したパスマッピング手法は与えられた RTL フォールスパスのうち，ほとんどすべてをマッピングできたと言える．

2.6. RTL フォールスパス情報を保存する論理合成手法

パスマッピング法を提案したことにより，現在のところ，RTL フォールスパスをゲートレベルにマッピングする手法は 2 つある．1 つはモジュール境界保存論理合成 (MIP-LS)[10] を利用すること，もう 1 つはモジュールの境界をまたぐ論理合成 (MIB-LS) を用いて合成した回路に対して，2.3 節で提案したパスマッピング法を利用することである．モジュール境界保存論理合成を用いる場合，与えられたすべての RTL フォールスパスをゲートレベルで利用可能であるという利点があるが，論理最適化をモジュールの境界を超えて行うことができず，面積，遅延オーバーヘッドが大きくなるという問題がある．また，提案したパスマッピング法を用いると任意の論理合成を用いることができるため，面積，遅延オーバーヘッドが存在しないが，論理最適化の結果，RTL パスに対応するゲートレベルパスが存在しなくなり，マッピングできない RTL フォールスパスが存在した．

本節では，低いオーバーヘッドを実現しながら，RTL パスとゲートレベルパスの完全なマッピングを保証する**マッピング箇所保存論理合成** (MPP-LS) を提案する．表 2.6 にこれらの論理合成の特徴を示す．RTL 回路中のすべてのモジュール境界を保存するモジュール境界保存論理合成とは異なり，提案手法では他のパスと区別するために必要十分なマッピング箇所のみを保存する．回路を論理合成する前に RTL フォールスパスが得られた場合，提案手法ではフォールスパスと他のパスを区別するために必要十分なマッピング箇所のみを保持するため，さらにマッピング箇所を削減することが可能となる．一方で，RTL パスとゲートレベルパスの完全な対応が保証できない任意の論理合成とは異なり，提案手法では完全なマッピングを保証する．

2.6.1 マッピング箇所保存論理合成

設計フローにおける IP の実装や，Engineering change を行うために，市販の論理合成ツールは，特定の RTL 信号線をゲートレベル回路に保存する機能を有している．例えば，Synopsys 社の DesignCompiler においては，RTL 信号線を保持するにために，“set_dont_touch” コマンドが利用可能である．この機能を利用することで，マッピング箇所保存論理合成を実現することを考える．

マッピング箇所保存論理合成を用いることで，RTL パスを構成する各ビットスライス RTL 信号線と機能等価なゲートレベル信号線を得ることができる．そのため，パスマッピング手法の手順 1-2 が不要となるため，信号線マッピングの繰り返しが不要となる．

2.6.2 RTL パスを識別する最小の RTL 信号線集合の抽出

ゲートレベルにマッピングする対象の RTL パス集合が与えられたとき，RTL パス集合を一意に識別する RTL 信号線のみをマッピングすることで，マッピング箇所を削減することができる．本副節では，与えられた RTL パス集合を一意に識別する最小の RTL 信号線集合を得る手法を提案する．

まず，RTL セグメント，再収斂セグメント対，ファンインセグメント対，ファンアウトセグメント対をそれぞれ，以下のように定義する．

定義 16 (RTL セグメント) *RTL* 信号線の順序集合 $p^R = \{e_1^R, \ldots, e_n^R\}$ を **RTL セグメント**と呼ぶ．ただし，$e_i^R (2 \leq i \leq n-1)$ は e_{i-1}^R を入力として持つモジュールと e_{i+1}^R を出力として持つモジュール間を接続する *RTL* 信号線である． □

特に，e_1^R が外部入力もしくはレジスタの出力に隣接しており，e_n^R が外部出力もしくはレジスタの入力に隣接する RTL セグメント p^R は RTL パスである．

定義 17 (再収斂セグメント対) 以下の条件を満たすセグメント $p^R = \{e_1^R, \ldots, e_n^R\}$ と $q^R = \{d_1^R, \ldots, d_m^R\}$ の対を**再収斂セグメント対**と呼ぶ．

1. $e_1^R = d_1^R$
2. $e_n^R = d_m^R$ □

定義 18 (ファンインセグメント対) 以下の条件を満たすセグメント $p^R = \{e_1^R, \ldots, e_n^R\}$ と $q^R = \{d_1^R, \ldots, d_m^R\}$ の対を**ファンインセグメント対**と呼ぶ．

1. e_1^R と d_1^R は外部入力またはレジスタの出力と隣接する *RTL* 信号線
2. $e_n^R = d_m^R$
3. $e_i^R \neq d_j^R (1 \leq i < n, 1 \leq j < m)$ □

定義 19 (ファンアウトセグメント対) 以下の条件を満たすセグメント $p^R = \{e_1^R, \ldots, e_n^R\}$ と $q^R = \{d_1^R, \ldots, d_m^R\}$ の対を**ファンアウトセグメント対**と呼ぶ．

1. $e_1^R = d_1^R$
2. e_n^R と d_m^R は外部入力またはレジスタの出力と隣接する *RTL* 信号線
3. $e_i^R \neq d_j^R (1 < i \leq n, 1 < j \leq m)$ □

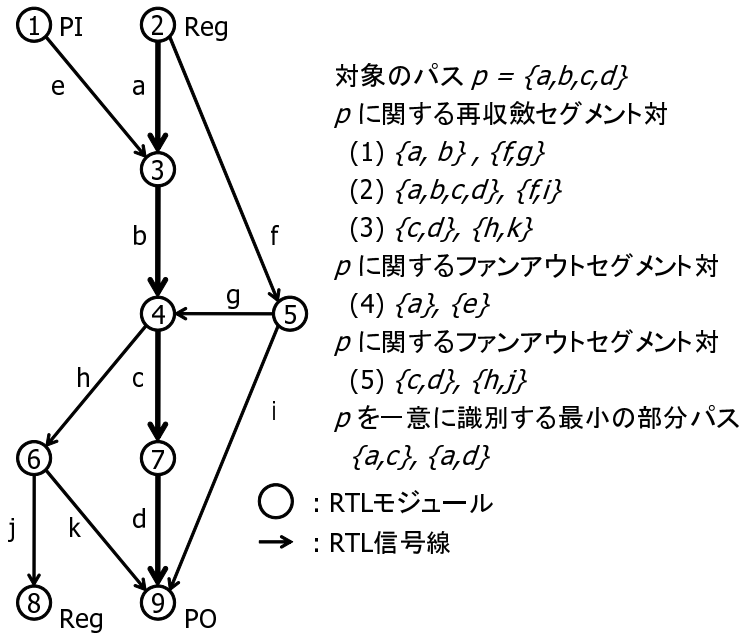

図 2.4 RTL パスの一意性

図 2.4 に示す回路のトポロジグラフ中の RTL パス $p = \{a, b, c, d\}$ を考える．p を一意に識別する最小の部分 RTL パス集合は，再収斂セグメント対，ファンインセグメント対，ファンアウトセグメント対の要素によって構成される．

図 2.4 の例では，p に関する再収斂セグメント対は，(1) $\{a, b\}$ と $\{f, g\}$，(2) $\{a, b, c, d\}$ と $\{f, i\}$，(3) $\{c, d\}$ と $\{h, k\}$ であり，ファンインセグメント対は (4) $\{a\}$ と $\{e\}$ であり，ファンアウトセグメント対は (5) $\{c, d\}$ と $\{h, j\}$ である．

RTL パス集合 P を識別するためには，対象パスに関する各セグメント対の信号線を少なくとも 1 つ選ぶ必要がある．このケースでは，p を一意に識別する RTL パスは，(1) $\{a, b\}$, (2) $\{a, b, c, d\}$, (3) $\{c, d\}$, (4) $\{a\}$,(5) $\{c, d\}$ の各セグメントの要素を少なくとも 1 つ含む必要がある．これは，最小被覆問題に帰着することができ，この例での p を一意に識別する部分パスは $\{a, c\}$ もしくは $\{a, d\}$ である．

RTL パス集合が与えられたとき，その集合を一意に識別する最小の RTL 信号線の集合は，以下のように得ることができる．まず，RTL パスの集合に含まれる各 RTL パスに関する，再収斂セグメント対，ファンインセグメント対，ファン

表 2.7 RTL ベンチマーク回路の回路特性

	#Bits	#PIs	#POs	#Regs	#OPs	#MUXes
LWF	16	2	2	5	3	7
Tseng	8	3	2	6	7	5

アウトセグメント対をそれぞれ列挙する．次に，列挙したセグメント対の少なくとも１つの信号線を被覆するよう，最小個の RTL 信号線集合を最小被覆問題を解くことで得る．

2.7. マッピング箇所保存論理合成の評価実験

実験結果では，再収斂するパスが存在する 2 つの RTL ベンチマーク回路，LWF と Tseng について，提案した論理合成手法を評価する．表 2.7 に LWF と Tseng の回路特性を示す．ただし，“#Bits” はビット幅，“#PIs” は外部入力数，“#POs” は外部出力数，“#Regs” はレジスタ数，“#OPs” は演算器数，“#MUXes” はマルチプレクサ数を示す．

また，表 2.8 に各回路の RTL パス数を示す．ただし，“#Paths” の下の “Total” は総パス数，“DP” はデータパス中のパス数，“Ctrl” はコントローラ中のパス数を示す．“#False” は文献 [10] の手法で判定されたデータパス中の RTL フォールスパス数を示す．“#MP w/o uniq.” はパスの一意性を考慮しない時のマッピング箇所の数，“#MP w/ uniq.” はパスの一意性を考慮したときのマッピング箇所の数を示す．その下の “All” はすべての RTL パスについて，各パスをマッピングする際に必要となるマッピング箇所の数，“False” は与えられたフォールスパスをマッピングする際に必要となるマッピング箇所の数を示す．ここで，パスの一意性を用いた最小のマッピング箇所を算出したとき，最小となるマッピング箇所の組合せ方法が複数存在することがある．“#Comb” はその組合せの数を示す．表 2.7 より，パスの一意性を用いることで保持すべきマッピング箇所を多く削減できることが分かる．さらに，RTL フォールスパスのみをマッピングする場合，マッピング箇所の数をさらに削減できる．

表 2.8 パス数とマッピング箇所の数

	#Paths			#False	#MP w/o uniq.		#MP w/ uniq.			
							All		False	
	Total	DP	Ctrl		All	False		#Comb		#Comb
LWF	45	19	26	5	14	8	3	4	1	2
Tseng	62	20	42	6	23	13	2	6	1	3

実験では論理合成ツールとして Synopsys 社の Design compiler(C-2009.06) を用い，使用した計算機は，Sun Microsystems 社の SunFire X4100 (Dual Opteron 256, 3.0GHz, 16GB) を用いた．論理合成では，制約無し論理合成 (MIB-LS)，モジュール境界保存論理合成 (MIP-LS)，および，提案手法 (MPP-LS) をそれぞれ DesignCompiler のオプションを変更することで実現した．MIB-LS を実現するためには，DesignCompiler のオプションに制約を与えずに合成を行った．MIP-LS を実現するためには，2 種類の方法で合成を行った．1 つは，RTL の各モジュールをそれぞれ合成し，ゲートレベルでそれらを接続する論理合成を行った．つまり，この合成を行うことで，RTL モジュール間の相互接続がゲートレベルにおいて変更されないことが保証できる．この合成方法を単に “MIP-LS” と呼ぶ．もう一方は，“compile_ultra” コマンドの “no_boundary_optimization”，“no_autoungroup” の各オプションをそれぞれ指定した．この合成方法を “MIP′-LS” と呼ぶ．MPP-LS を実現するためには，保持する対象の信号線に対して “set_dont_touch” コマンドを利用することでマッピング箇所を指定し，合成を行った．

表 2.9 に RTL ベンチマーク回路に対して MIB-LS，MIP-LS，MPP-LS，MIP′-LS をそれぞれ適用し，得られたゲートレベル回路の回路特性を示す．LWF に対しては 2 つのタイミング制約（40ns と 10ns）を用い，この制約の下で MIB-LS，MPP-LS，MIP-LS，MIP′-LS をそれぞれ適用した．ただし，“面積” はゲート数 (NAND=1)，“遅延” はクロック周期 (ns) を示す．提案手法である MPP-LS におけるマッピング箇所の組合せについては，すべて試行し面積が一番小さくなったものを実験結果として用いている．また，“#Paths” の右の “Total” は総パス数，“Datapath” はデータパス中のパス数，“False” はデータパス中のフォールスパス数，“%Ratio” はデータパス中のパス数と得られたフォールスパス数の比である

表 2.9 LWF の論理合成結果 (タイミング制約 40ns)

合成手法		MIB-LS	MPP-LS	MIP-LS	MIP′-LS
面積		664	709	784	799
遅延		33.81	37.13	38.23	39.16
#Paths	Total	29,098	46,838	47,446	47,446
	Data path	11,072	13,420	14,404	12,076
	False	3,296	4,272	4,574	3,798
	%Ratio	29.8	31.8	31.8	31.5

表 2.10 LWF の論理合成結果 (タイミング制約 10ns)

合成手法		MIB-LS	MPP-LS	MIP-LS	MIP′-LS
面積		1,063	1,096	796	1,092
遅延		9.20	9.15	*(30.37)*	9.20
#Paths	Total	138,262	111,510	59,086	123,878
	Data path	52,684	38,924	14,404	38,492
	False	18,328	14,952	4,574	14,824
	%Ratio	34.8	38.4	31.8	38.5

フォールスパス判定率を示す．同様に，Tseng に対してタイミング制約 20ns および 10ns を用いて合成を行った．実験結果より，MPP-LS の面積は MIB-LS とほぼ同等であった．LWF の 10ns のタイミング制約，および，Tseng の 20ns，10ns のタイミング制約について MIP-LS はそれらの制約を満たす回路を得ることができなかった．すべての回路と制約条件下において，MPP-LS のフォールスパス判定率はタイミング違反を起こすケースを除いて最も高い結果となった．

最後に，速度マージンに対する影響を調べるため，各手法で合成された回路のパススラックの分布図を示す．図 2.5 と図 2.6 に LWF と Tseng をそれぞれ 10ns のタイミング制約の下論理合成した回路のパススラックの分布図を示す．ただし，タイミング制約に違反した MIP-LS の結果はこれらの図に掲載していない．各グラフの x 軸はパスのスラックの範囲 (ns)，y 軸はその範囲に存在するパス数を示

表 2.11 Tseng の論理合成結果 (タイミング制約 20ns)

合成手法		MIB-LS	MPP-LS	MIP-LS	MIP′-LS
面積		699	726	860	868
遅延		19.02	19.07	*(20.86)*	14.41
#Paths	Total	33,932	33,940	36,152	34,258
	Data path	30,736	30,680	32,490	30,868
	False	1,072	15,906	15,966	15,996
	%Ratio	3.5	51.8	49.1	51.8

表 2.12 Tseng の論理合成結果 (タイミング制約 10ns)

合成手法		MIB-LS	MPP-LS	MIP-LS	MIP′-LS
面積		958	935	859	1,057
遅延		9.20	9.16	*(20.86)*	9.20
#Paths	Total	74,344	68,036	36,152	65,658
	Data path	68,204	64,088	32,490	61,820
	False	3,568	14,574	16,404	12,542
	%Ratio	5.2	22.7	50.5	20.3

す．Tseng に対しては，MIP′-LS が速度マージンに対して最も大きな影響を与えており，提案手法である MPP-LS ではその影響を緩和することができた．LWF に対しては，1.2ns 付近ですべて立ち上がっており，各論理合成手法間に差異はない．

2.8. 結言

RTL 回路から得られる情報を，論理合成後のゲートレベル回路で利用することができれば，高位の設計情報（RTL や高位合成情報）を用いたテスト手法に有用である．本論文では，RTL 回路中のパスとゲートレベル回路中のパスの対応関係に着目した．この対応関係の存在により，RTL パスを用いることで，多数の

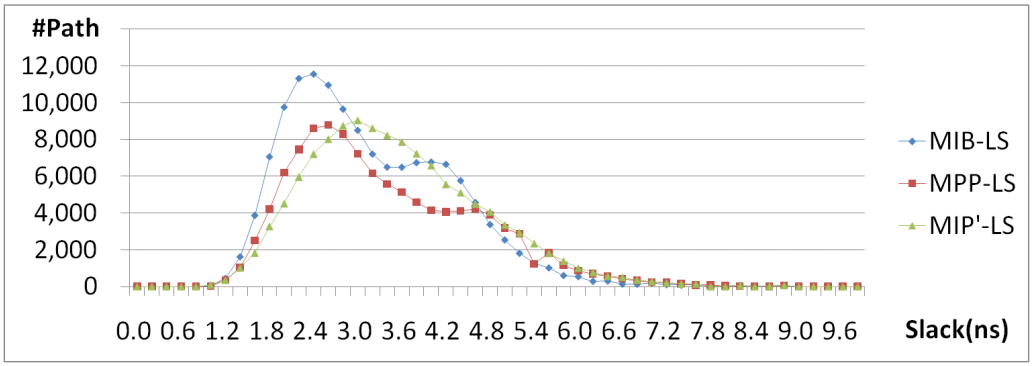

図 2.5 LWF のパス長の分布図 (10ns)

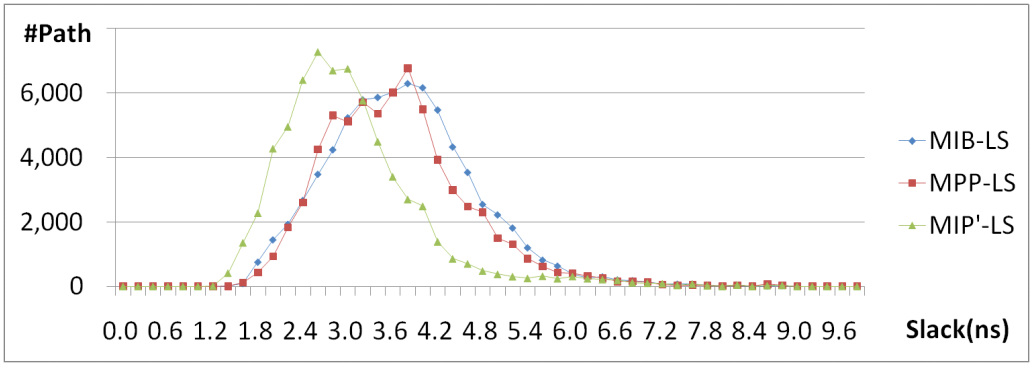

図 2.6 Tseng のパス長の分布図 (10ns).

パスが存在し，扱いが難しいゲートレベルパスを容易に扱うことが可能となる．例としては，RTL 設計情報を用いて高速に RTL フォールスパスを判定する手法 [10, 18] への応用が可能となる．現在のところ，RTL パスとゲートレベルパスの対応関係を得るためには論理合成に制約を置く必要があった．

本章では，論理合成に制約を置かずに，RTL パスの集合とゲートレベルパスの集合の対応関係を得る手法を提案した．これは RTL パスのゲートレベルパスへのマッピングを行う初めての手法であり，文献 [10] の手法で判定された RTL フォールスパスがゲートレベルフォールスパスにマッピングされることを示した．実験結果では，提案したパスマッピング手法をフォールスパスマッピングに応用し，

論理合成に制約を置かずに合成したゲートレベル回路中から多くのゲートレベルフォールスパスを得ることができた．今後の課題としては，パスマッピング率を向上させるために，ビットスライス RTL 信号線に対応する “複数” のゲートレベル信号線を扱うことが挙げられる．

また，RTL 回路中のパスをゲートレベル回路にマッピングする，パスマッピングを行うにあたり，完全なパスマッピングを実現する**マッピング箇所保存論理合成**手法を提案した．実験結果により，提案手法はモジュール境界保存論理合成を用いるよりも面積，遅延オーバーヘッドを緩和し，パスマッピング法を用いるよりもフォールスパス情報を RTL からゲートレベルへ伝搬する能力が向上したことを示した．さらに，速度マージンへの影響を調べ，提案手法はその影響を緩和したことを示した．今後の課題としては，さらなる面積，遅延オーバーヘッドの削減のために，最適なマッピング箇所の組合せ方法を提案することが挙げられる．

第3章　非同期式回路のテスト

3.1. 緒言

大域非同期局所同期 (Globally-Asynchronous, Locally-Synchronous: GALS) システムは，半導体の微細化，大規模化に伴う大域クロックの分配の難しさを解決する現実的な手段として知られている．そのため，非同期式回路に対しても，同期式回路と同等の高いテスト品質が求められる．GALS システムでは局所的に同期された機能モジュールが，非同期式のハンドシェイクプロトコルを用いて通信を行う．GALS 設計手法を用いることで既存の機能モジュールの再利用や，モジュール間のタイミング設計が容易となる．さらに，様々な種類のクロックを利用することができるため，各モジュールについて最適なクロックを利用することで，システム全体の消費電力を低く抑えることが可能である．ITRS では 2015 年には回路のうち 25% が，2024 年には 49% がハンドシェイクプロトコルを用いて駆動されることが予測されている [1].

非同期式回路に対して同期式回路と同じ DFT 手法を適用することを考えた場合，非同期式回路中に存在するすべての組合せループを切るように，レベルセンシティブスキャン設計 (Level Sensitive Scan Design: LSSD) で用いるスキャン素子を挿入する手法がある [3]．組合せループを切ることで，組合せ回路部分に対して組合せテスト生成手法を適用することができ，同期式回路に対するテスト手法と同等のテスト品質を保証することができるが，多くの LSSD スキャン素子を挿入する必要があり，高い面積，遅延オーバーヘッドを招く問題がある．この問題を解決するために，複数のスキャン設計手法が提案されており，それらは部分スキャン法と完全スキャン法に分類することができる．部分スキャン法 [19, 20] では，組合せ回路ループを切る数を削減し，核回路をテスト容易な順序回路として

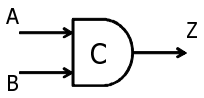

図 3.1 2 入力対称 C 素子

表 3.1 2 入力対称 C 素子の真理値表

A	B	Z
0	0	0
0	1	Q
1	0	Q
1	1	1

残す．文献 [19] では，CHAIN コミュニケーションチャネル回路に対して，大域フィードバックループのみを切る手法を提案している．文献 [20] では，回路グラフの最小フィードバック辺集合を考慮し，ループを切る箇所を最小化する手法が提案されている．

文献 [21, 22, 23] では，LSSD のスキャン素子を用いた手法のオーバーヘッドを削減する完全スキャン設計手法が提案されている．同期式回路に対する LSSD のスキャン素子を用いた完全スキャン設計のオーバーヘッドを削減する手法として，L1L2*スキャン設計法 [24, 25] がある．L1L2*スキャン設計法では，シングルラッチ設計された回路に対して完全スキャン設計を実現する手法であり，LSSD のスキャン素子を用いて完全スキャン設計を実現する場合に比べ面積，遅延オーバーヘッドを抑制することができる．Beest ら [21] は，非同期式回路に対して L1L2*スキャン設計を行うために，組合せループをシングルスキャンラッチで切り，スキャンチェーン上の連続する 2 つのスキャンラッチをスキャンシフトレジスタラッチ (スキャン SRL) として扱う手法を提案している．LSSD では 1 つのスキャン素子に対して 2 つのラッチが必要だったのに対し，L1L2*スキャン設計では追加するラッチの数を半分に削減することが可能となる．さらに面積，遅延オーバーヘッドを抑制する手法として，マルチプレクサベースのスキャン C 素子を用いた

L1L2*スキャン設計法 [22] を提案している．C 素子 (図 3.1 参照) は入力 A と入力 B と現状態 Q の多数決の結果を出力する，非同期式回路で多く用いられる非同期式順序素子であり，表 3.1 の真理値表を持つ．マルチプレクサベースのスキャン C 素子は，C 素子内部の組合せループに追加のマルチプレクサを追加し，機能パスにはラッチを挿入しない．Shi ら [23] はマルチプレクサベースのスキャン C 素子を応用し，遷移故障のテスト手法を提案している．

非同期式回路に対して完全スキャン設計を行う DFT 手法を用いる最大の利点は，開発が進んでいる組合せ回路用のテスト生成手法を応用できることである．実用的な非同期式回路を設計するためには，自動設計フローが不可欠であり，非同期式回路設計を可能とする複数の CAD ツール (Petrify [26]，Balsa [27] など) が開発されている．しかし，非同期式回路をテストする一般的な CAD ツールは現在のところ，開発されていない．そのため，市販の同期式回路向けに開発された ATPG ツールを利用できることは大きな利点となる．一方で，非同期式回路に対して完全スキャン設計を行うことで，非同期式回路の利点の 1 つであるクロックが存在しない事によるクロックツリー構築が不要であること，テスト時における消費電力の増加などの問題が生じる．しかし，非同期式回路に対して完全スキャン設計を行ったとしても同期式動作を行うのはテスト時のみであり，通常動作時には非同期式回路を用いる利点をそのまま享受することができる．また，完全スキャン設計を適用した非同期式回路は，研究，開発が進んでいる同期式回路に対するクロックツリーの構築手法，テスト時の消費電力削減手法が適用可能であるため，同期式テスト手法を用いる際に生じる問題を抑制しながら，高いテスト品質を得ることができる．

L1L2*スキャン設計を用いた手法 [21, 22] はオーバーヘッドが少ないという利点があるが，**キャプチャ衝突**，**組合せループの存在**という 2 つの問題により，非同期式回路の組合せ回路部分にさえ完全な故障検出効率を保証することができない．具体的には，C 素子の状態遷移はその状態に依存するため，テスト応答を C 素子に取り込むためには，C 素子の状態を適切な値に設定しておく必要がある．例えば，C 素子の 2 つの入力に 1/0（誤りを含む値）と 0 が入力されたとき，誤りを C 素子に取り込むために必要な C 素子の内部状態は 1 に設定する必要があ

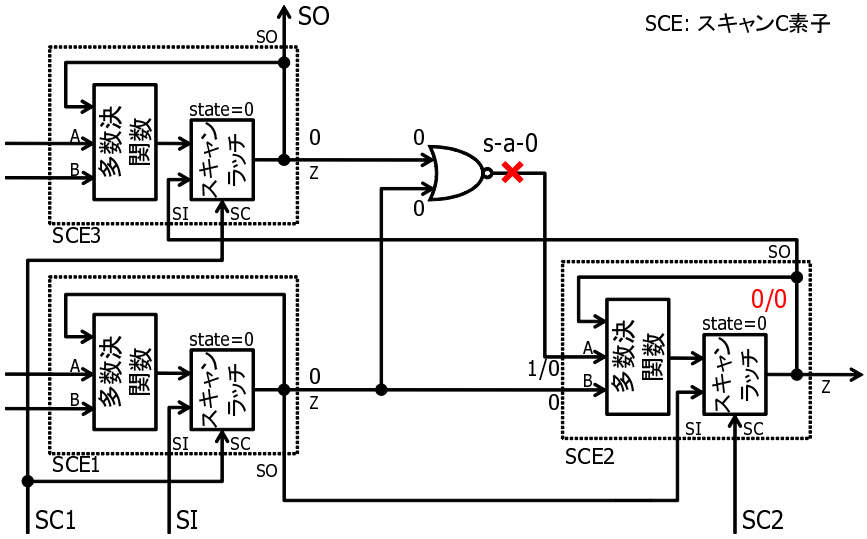

図 3.2 スキャン C 素子でのキャプチャ衝突

る．図 3.2 は文献 [21] で提案されているスキャン素子を用いて DFT 下回路において，誤りがスキャン C 素子の内部状態によって取り込まれない例である．スキャンチェーン上の前のスキャン C 素子 (SCE1) および，後のスキャン C 素子 (SCE3) の両方が同じ値 0 を持つため，SCE2 の内部状態は 0 となり，SCE2 の入力に伝搬されてきた誤り 1/0 を取り込むことができない．この状況を**キャプチャ衝突**と呼ぶことにするが，文献 [21, 22] ではこの問題について記述されておらず，テスト応答を取り込めないスキャン C 素子が存在することにより，組合せ回路部分に完全なテストを保証できないという問題がある．

もう一方の問題である**組合せループの存在**については，図 3.3 のように，L1L2* スキャン設計においてマルチプレクサベースのスキャン C 素子が利用されたとき，1 つのスキャンラッチしか含まないループが存在する可能性がある．そして，スキャンラッチがテスト応答を取り込む際にそのループは組合せループとして動作し，そのループ上に存在する否定素子の数が奇数であれば組合せループ上で発振し，テスト応答を取り込むことができない．

本章では非同期式回路に対して上述の問題を解決する L1L2*完全スキャン設計法を提案する．提案手法は以下の 3 つの特徴を持つ．

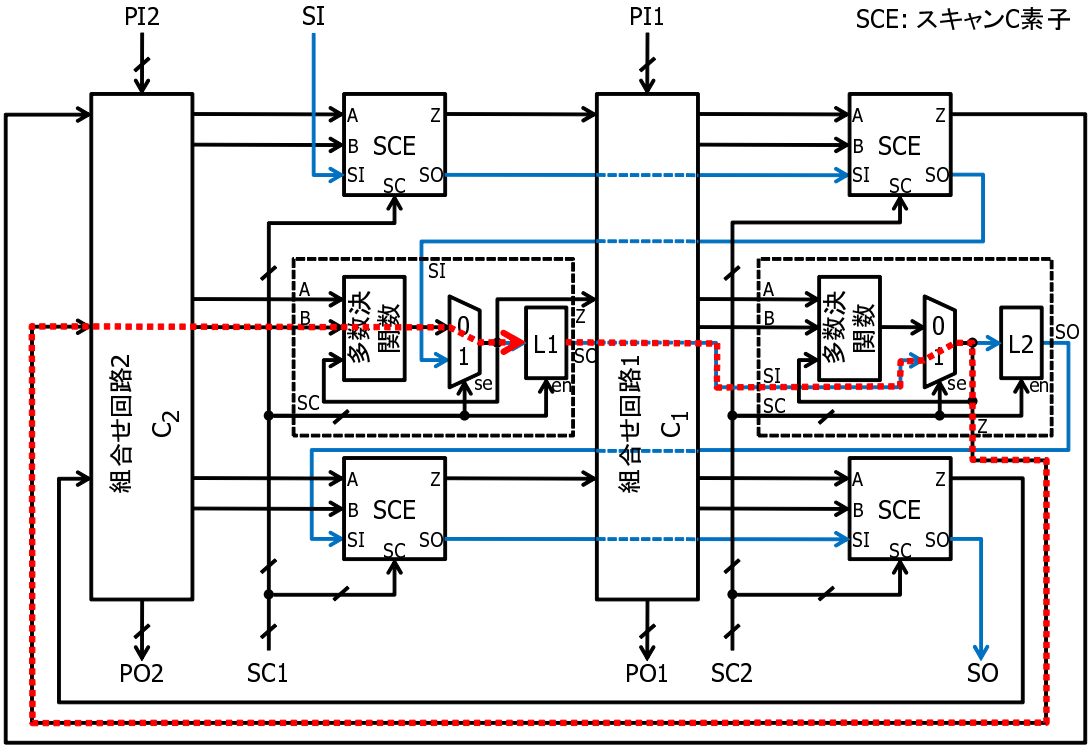

図 3.3 マルチプレクサベースのスキャン C 素子を用いた回路

1. 既存の L1L2*完全スキャン設計手法で達成できない，非同期式回路の組合せ回路部分に対する完全なテストを実現

2. 既存の完全スキャン設計法で考慮されていない，順序回路部分に対する完全なテストを実現

3. オーバーヘッドの面で一番優れる文献 [22] の手法と同等のオーバーヘッド

本章の構成は次のとおりである．まず，諸定義を 3.2 節で行う．3.3 節において，キャプチャ衝突を防ぎ，組合せ回路部分に対して完全なテストを保証する DFT 手法を提案する．3.4 節では組合せループの問題を解消し，順序回路部分に対して完全なテストを保証しながら，オーバーヘッドは既存の手法と同等で実現可能なスキャン素子を提案する．2.5 節では実験結果について述べ，最後に，3.6 節で本章の結言を述べる．

3.2. 諸定義

3.2.1 回路モデル

本章では順序素子として 2 入力の C 素子を用い，組合せ回路部分には閉路が存在しない非同期式順序回路を対象とする．C 素子は非同期式回路で最も一般的に用いられる順序素子であり，特に，ヌルコンベンショナルロジック (NCL)[28] 設計手法などを用いて設計された非同期式回路である，ディレイインセンシティブ (DI) 回路は C 素子が順序素子として利用されている．また，スキャン C 素子は通常動作とスキャン動作の 2 つのモードで動作し，スキャン動作では，ロード機能とホールド機能を持つラッチとして動作する．

3.2.2 故障モデル

本章では非同期式回路を組合せ回路部分と個々の順序素子に分けてテストすることを考える．組合せ回路部分に対しては，入力変数を変化させず，組合せ回路の出力の論理関数を他の論理関数に変更する検出可能な論理故障を対象とする．組合せ回路中のすべての検出可能な論理故障を検出したとき，組合せ回路は**完全にテストされた**という．

同様に，順序素子に対しては入力変数と状態変数を変化させず，順序素子の状態表を他の状態表に変更する検出可能な論理故障を対象とする．順序素子中のすべての検出可能な論理故障を検出したとき順序素子は完全にテストされたという．

3.2.3 2 部非同期式回路構造

非同期式回路に対して，L1L2*スキャン設計を適用可能なシングルラッチ設計の回路構造を 2 部非同期式回路構造として以下のように定義する．

定義 20 (2 部非同期式回路構造) 非同期式回路 C について，C の組合せ回路部分が *2* つの互いに素な部分回路 C_1 と C_2 に分けられ，C の順序素子の集合が *2* つの部分集合 *L1* と *L2* に分割されるとき，C は **2 部非同期式回路構造**である．ただ

し，*L1* は C_1 のみを駆動し，C_2 によってのみ駆動され，*L2* は C_2 のみを駆動し，C_1 によってのみ駆動される．□

次に，2 部非同期式回路構造を持つ非同期式回路に対して，L1L2*完全スキャン設計を行った際に必要となるスキャンパスを 2 部スキャンパスとして以下のように定義する．

定義 21 (2 部スキャンパス) スキャン素子の系列であるスキャンパス p に対して，以下の条件を満たす p を **2 部スキャンパス** *(B-scan パス)* と呼ぶ．

1. p 上のスキャン素子は *2* つの部分集合から成る
2. p 上の任意の連続するスキャン素子は異なる集合に属する
3. 各集合のすべてのスキャン素子は同時に同じ動作を行うことができる □

次に，組合せ回路に対して完全なテストを保証する，2 部完全スキャン可検査性を定義するために必要な透過ファンイン，S グラフ，無向閉路を以下のように定義する．

定義 22 (透過ファンイン) 信号線 l から他の信号線 m へ組合せ回路要素のみを通る接続があるとき，l は m の**透過ファンイン**と呼ぶ．□

定義 23 (S グラフ) 非同期式回路 C の S グラフ $G = (V, E)$ は，有向グラフであり，頂点 $v \in V$ は C の順序素子を表し，辺 $(u, v) \in E$ は u の出力は v の入力の透過ファンインであることを示す．□

定義 24 (無向閉路) すべての辺の方向を無視した S グラフ中の閉路を S グラフ上の**無向閉路**と呼ぶ．□

3.3. 2 部完全スキャン設計

本節で対象とするスキャン C 素子は，文献 [21] で提案されているシングルラッチタイプのスキャン C 素子か，3.4 節で提案するスキャン C 素子を考える．つまり，提案する完全スキャン設計手法においてはどちらのスキャン C 素子も利用可能である．

3.3.1 2部完全スキャン可検査性

キャプチャ衝突を防ぐために，非同期式回路に対して，組合せ回路部分の完全なテストを保証する可検査性を提案する．前述のように，あるC素子において誤りを取り込むためには，C素子に適切な値を設定する必要がある．この値を**キャプチャパターン**と呼ぶ．スキャンC素子s_iにおいて出力応答を取り込むことを考えたとき，s_iとスキャンシフトレジスタラッチを構成するスキャンラッチとして，前のスキャン素子s_{i-1}もしくは次のスキャン素子s_{i+1}のどちらかを選ぶことができる．提案する可検査性では任意の連続する2つのスキャンC素子s_iとs_{i+1}の間には，組合せ回路を通した接続が存在しない，もしくはs_{i+1}とs_iの間に組合せ回路を通した接続が存在しないことが必要となる．この条件により，故障を検出するテストパターンと，対応するキャプチャパターンを独立に制御可能となる．

定義 25 (2部完全スキャン可検査性) 非同期式回路Cに対して，以下の条件を満たすCは**2部完全スキャン可検査性** *(BF-scan*可検査性*)*を満たすという．

1. Cのすべての順序素子はスキャン素子である
2. Cは少なくとも*1*つのスキャンパスを持ち，各順序素子はいずれかのスキャンパスに含まれる
3. Cは組合せ部分回路C_1とC_2，順序素子の集合*L1*と*L2*から構成される*2*部非同期式回路構造を持つ
4. 各スキャンパス$p = s_1, \ldots, s_n$は順序素子の集合*L1*と*L2*で構成される*B-scan*パスである
5. p上の*2*つの連続するスキャン素子s_iとs_{i+1} $(i = 1, \ldots, n-1)$について，s_{i+1}の出力はs_iの入力の透過ファンインでない，もしくはs_iの出力はs_{i+1}の入力の透過ファンインでない □

条件4. および5. を満たすスキャンパスpを**2部完全スキャンパス** (BF-scanパス) と呼ぶ．

定理 3 非同期式回路 C について，C が *BF-scan* 可検査性を満たすなら C の組合せ回路部分は完全にテストできる．

証明　$C_1, C_2, L1, L2$ をそれぞれ，定義 25 の条件 3. に示す部分回路と順序素子の集合とする．f を C の検出可能な論理故障とすると，C の組合せ回路部分のいずれかの出力 o に誤りを伝搬させるテストパターン t が存在する．一般性を失わないように，スキャンパス上の s_i の次のスキャン素子 s_{i+1} から s_i に対する組合せ接続が存在しない，もしくは，s_i がスキャンパス上の最後のスキャン素子であるものとする．この条件下で，s_i と s_{i+1} を用いてスキャンシフトレジスタラッチを構成するとき，他のスキャンシフトレジスタラッチは s_i とは独立に制御可能である．f を検出するテストパターン t に対して，特定の値を設定する必要があるのは o の透過ファンインにのみ存在し，s_{i+1} は o の透過ファンインではない．よって，t とキャプチャパターンは独立に制御可能である．さらに，B-scan パスの性質を用いることで，取り込まれた出力応答はスキャンアウトまで伝搬可能である．よって定理は証明された．　□

3.3.2 組合せ回路に対するテスト生成手法

組合せ回路に対するテストを考えたとき，まず，組合せ回路中の検出可能な論理故障 f を検出するテストパターンを生成することを考える．ただし，誤りはスキャン C 素子 s_i によって取り込むとする．キャプチャパターンは s_i の 2 つの入力に伝搬される期待値と誤りの少数決で決定することができる．また，期待値と誤りは f に対して生成されたテストパターンを用いて故障シミュレーションを行うことで得られる．少数決の結果が等しく，s_i の 2 つの入力に同じ期待値と誤りが伝搬された場合は，キャプチャパターンとは独立に誤りを取り込むことができる．以上の条件に合致するテストパターンを生成できない場合，f は冗長である．

このテストパターンは市販の組合せ ATPG を用いて生成することができ，キャプチャパターンは市販の故障シミュレーションツールを用いて得ることが可能である．また，テスト実行時間を削減するために，テストパターンの数を以下のように削減できる．1 故障に対するテストパターンとキャプチャパターンの対を用

いることで，複数の故障が検出される可能性がある．これは故障シミュレーションツールを用いて確かめることが可能であり，検出可能であるとわかった故障については故障リストから取り除くことができる．さらに，複数のテストパターンとキャプチャパターンの対は，それらが両立可能であれば同時に印加できる．

3.3.3 提案する DFT 手法

本副節では任意の非同期式回路を BF-scan 回路構造に設計変更する DFT 手法を提案する．まず，提案する DFT 手法を以下のように定式化する．

定義 26 (BF-scan 可検査 DFT)

入力: 非同期式回路 C とスキャンパスの数

出力: C と同じ機能を持つ *BF-scan* 可検査性を満たす非同期式回路

最適化目標: 追加するラッチの数の最小化 □

この問題を解くために，次の 3 つのステップから成るアルゴリズムを提案する．

1. C のすべての順序素子を対応するスキャン素子に取り替える
2. C の S グラフを $G_b = (V_1, V_2, E)$ の 2 部グラフに変換する
3. BF-scan パスを構築する

回路が 2 部グラフであるためには，S グラフの各無向閉路の長さが偶数であればよい．手順 2 において，S グラフを 2 部グラフに変換するために最小個の追加ラッチをスキャン素子の出力に挿入する．通常動作への影響を最小化するために，次の整数計画問題を解くことで追加ラッチを挿入するスキャン素子の数を最小化する．

Minimize:

$$\sum_{i=1}^{n} \delta_i$$

Subject to:

各無向閉路 c_j $(j = 1, \ldots, m)$ に対して，

$$\sum_{i=1}^{n} [(1 + \delta_i) \times IN_{i,j}] = 2 \times \epsilon_j$$

$$1 \leq \epsilon_j \leq n$$

ただし，n は S グラフの頂点数，δ_i $(i = 1, \ldots, n)$ は v_i の出力にラッチを付与するかどうかを表す論理変数，m は S グラフ中の無向閉路の数，ϵ_j は整数をとる変数，$IN_{i,j}(= 0, 1)$ は v_i が c_j に属していれば 1 になり，そうでない場合は 0 となる係数を示す．

手順 3 では，前の手順で得られた 2 部 S グラフを用いて BF-scan パスを構成する．まず，両立グラフを以下のように定義する．

定義 27 (両立グラフ) *2*部グラフ $G_b = (V_1, V_2, E_b)$ に対して，グラフ $G_c = (V, E_c)$ を G_b の**両立グラフ**と呼ぶ．ただし，V は $V_1 \cup V_2$ の頂点集合であり，$(u, v) \in E_c$ は G_b において v が u に隣接せず，v が V_2*(resp. V_1)* に属し，u が V_1 *(resp. V_2)* に属するとき，$u, v \in V$ を結ぶ辺である． □

ここで，2 部 S グラフ $G_b = (V_1, V_2, E_b)$ とその両立グラフ $G_c = (V, E_c)$ を用いて BF-scan パスを構築するアルゴリズムを示す．

1. 各スキャンパスについて，スキャンパスの先頭となるスキャン素子を選択する

2. 各スキャンパスについて，両立グラフ中で先に選択したスキャン素子に対応する頂点に隣接する頂点に対応するスキャン素子を，スキャンパス上の次のスキャン素子として選択する．選択可能なスキャン素子が存在しない場合は，回路に接続されていないスキャンラッチを新たに回路に付加し，両立グラフを更新する

3. すべての頂点を選択するまで 2. の手順を繰り返す

ただし，両立グラフの更新処理は，V に新しい頂点 v を付加し，前のスキャン素子が V_1(resp. V_2) に属していれば，v から V_1(resp. V_2) に属するすべての頂点への辺と，V_1(resp. V_2) に属するすべての頂点から v への辺を E_c に追加する．

BF-scan パスを構成する追加ラッチは通常動作の性能に影響を与えないので，この手順では追加するラッチの数は最小化していないが，大規模な非同期式回路に対しては，次のスキャン素子として選択可能な，独立なスキャン素子が十分存在するため、追加するラッチの数は少なくなると考えられる．

3.4. スキャンC素子とスキャンパスのテスト

3.4.1 スキャンC素子

本節ではC素子の値保持機能をスキャンパスのメモリ素子として扱うスキャンC素子を提案する．C素子の値保持機能を用いることで，提案するスキャンC素子は次の2つの特長を持つ．

1. スキャンパス上のすべてのC素子は連続する単一入力変化のパターンを用いて任意の状態遷移を発生することができるため完全なテストが可能である
2. ループに1つしかラッチが存在しない組合せループの問題を解消し，提案するスキャンC素子の遅延オーバーヘッドは今までで最も遅延オーバーヘッドが低いマルチプレクサベースの手法と同等である

図3.4に提案するスキャンC素子の実装例を示す．提案するスキャンC素子の構造は2つの部分，オリジナルのC素子とスキャン制御部から成る．この設計では，任意の種類のC素子が利用可能であり，スキャン制御部は表3.2に示す機能を持つ組合せ回路で実現する．スキャン制御部は次の3つの動作を行う．

1. 回路の通常動作とテスト中のキャプチャモードで用いる通常動作
2. C素子の内部状態を保持するホールド機能
3. 提案するスキャンC素子にSIの値を取り込むロード機能

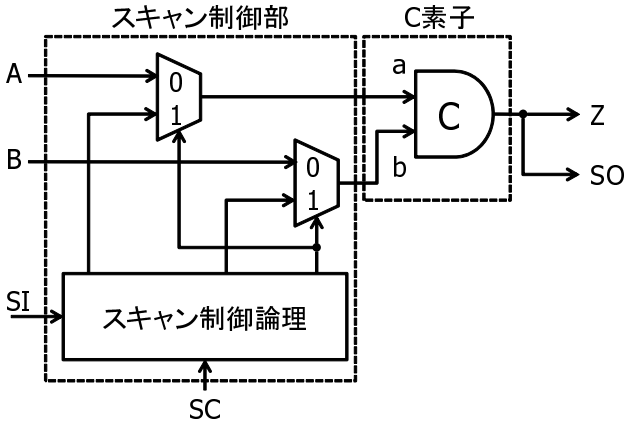

図 3.4 提案スキャン C 素子の実装

表 3.2 スキャン制御部の真理値表

SC	a	b	機能
00	A	B	通常動作 (キャプチャ)
01	0	1	スキャン動作 (ホールド)
10	1	0	スキャン動作 (ホールド)
11	SI	SI	スキャン動作 (ロード)

スキャン制御部の機能を以下に示す．機能 1. は SC=00 を印加することで，スキャン C 素子は通常動作に移行する．機能 2. は SC=01 を印加することで，C 素子に対して (a,b)=(0, 1) を印加し，SC=10 を印加することで (a,b)=(1, 0) を印加する．(a,b)=(0, 1)，(a,b)=(1, 0) のこの 2 つのパターンを**ホールドパターン**と呼ぶ．詳細は次の副節で述べるが，提案するスキャン C 素子を用いて出力応答を取り込む際，適切なホールドパターンを印加することでレースを回避することができる．機能 3. は SC=11 を印加することで，SI の値を C 素子に取り込む．

3.4.2 キャプチャ時のレース対策

提案するスキャンC素子では，ホールドパターンとして(a,b)=(0, 1)，(a,b)=(1, 0)のいずれかを印加することができる．ここで，内部状態を保持するために(a,b)=(0, 1)を印加することを考える．内部状態が1のC素子に対して，(A,B)=(1/0, 0)がC素子の入力に伝搬され，スキャン制御がホールドからロードに変わったとき，AがBよりも早く到達したとき内部状態は1/0になり，BがAよりも早く到達したとき内部状態は0になる．つまり，誤りによって内部状態が0になったのか，Bが早着したことにより0になったかが分からず，この取り込んだ応答を評価することができない．

しかし，(a,b)=(1, 0)を印加することを考えると，C素子の内部状態はA, Bどちらが早着するに関わらず1/0となり，誤りを取り込むことが可能である．よって，提案したスキャンC素子に伝搬してくる出力応答に基づいて適切なホールドパターンを選択することでレースを回避することができる．レースフリーのホールド動作に求められる適切なホールドパターンは，テスト生成時の故障シミュレーションによって決定することができる．

3.4.3 B-scanパス中のスキャンC素子のテスト

C素子の完全なテストは状態遷移図のすべての状態遷移を確かめることで実現することができる．つまり，安定状態から単一入力変化のパターンを印加し，その出力応答を観測する．ラッチも同様に2入力の順序素子であり，C素子と同様の手法で完全なテストが可能である．

図3.5に提案したスキャンC素子を4つ利用したB-scanパスを示す．B-scanパス上のC素子を完全にテストするためには，C素子を安定状態に設定し，C素子に対してテストパターンを印加して得られる任意の出力応答をSOで観測する必要がある．図3.6に2入力対称C素子の状態遷移図を示す．状態遷移図中には12個の遷移と6個の安定状態があり，C素子の遷移を変化なし(NOC), 出力更新(OF)，出力変化(OC)の3つに分類する．NOC遷移はC素子の内部値をホールドする遷移で，(a,b,c)=$(0, 0, 0) \rightarrow (0, 1, 0)$，$(0, 0, 0) \rightarrow (1, 0, 0)$，$(1, 1, 1) \rightarrow (0, 1, 1)$，

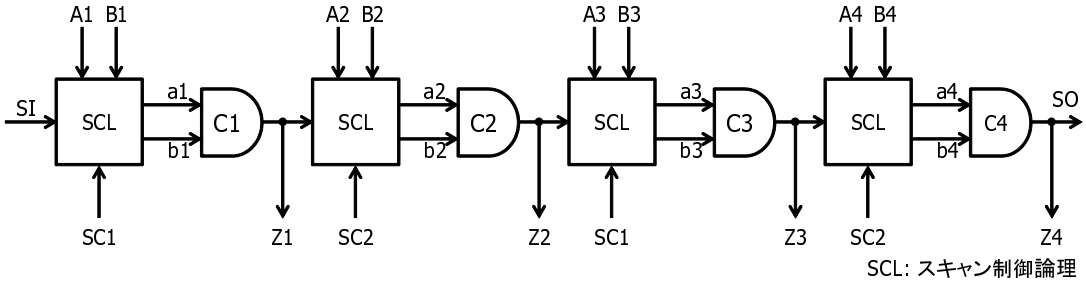

図 3.5 提案したスキャン C 素子を用いて構成した B-scan パス

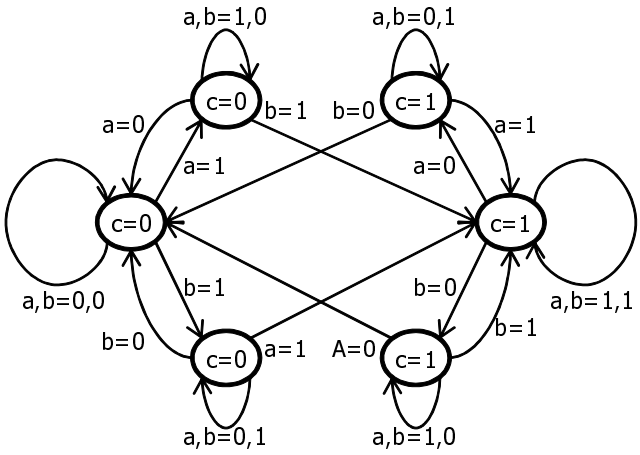

図 3.6 2 入力対称 C 素子の状態遷移図

$(1,1,1) \rightarrow (1,0,1)$ の 4 つの遷移が存在する．ただし，c は C 素子の内部値を示す．OF 遷移は C 素子の内部値と同じ値をロードする遷移であり，(a,b,c)=$(0,1,0) \rightarrow (0,0,0)$，$(1,0,0) \rightarrow (0,0,0)$，$(0,1,1) \rightarrow (1,1,1)$，$(1,0,1) \rightarrow (1,1,1)$ の 4 つの遷移が存在する．OC 遷移は C 素子の内部値と異なる値をロードする遷移であり，(a,b,c)=$(0,1,0) \rightarrow (1,1,1)$，$(1,0,0) \rightarrow (1,1,1)$，$(0,1,1) \rightarrow (0,0,0)$，$(1,0,1) \rightarrow (0,0,0)$ の 4 つの遷移が存在する．

NOC 遷移のうち $(0,0,0) \rightarrow (0,1,0)$ のテストは次の 2 つの手順で行う．(1) SC1=SC2=11，SI=0 を印加することで B-scan パス上のすべての C 素子を 0 で初期化する (2) SI=0 を印加しながら SC1=01，SC2=11 と SC1=11，SC2=01 を交

互に印加することでスキャンシフトを行う．ホールドパターン (a,b)=(0, 1) を L1 に印加することで，L1 のすべての C 素子に対して (0, 0, 0) → (0, 1, 0) の遷移が印加され，その応答が L2 でキャプチャされる．同様に，(a,b)=(0, 1) を L2 に印加することで，L2 のすべての C 素子に対して (0, 0, 0) → (0, 1, 0) の遷移が印加され，その応答が L1 でキャプチャされる．取り込まれた値は次のスキャン C 素子のスキャンインとして利用されるため，誤りは SO までシフトアウトされる．他の NOC 遷移についても，SI の値とホールドパターンの組合せを変更することで，同様にテストできる．

(a,b,c)=(0, 1, 0) → (0, 0, 0) の OF 遷移は以下の 3 つの手順を用いてテストすることができる．(1) SC1=SC2=11，SI=0 を印加することで B-scan パス上のすべての C 素子を 0 に初期化する．(2) SC1=SC2=01，SI=0 を印加することですべての C 素子に対してホールドパターン (a,b)=(0, 1) を印加する．(3) SC1=SC2=11，SI=0 を印加することですべての C 素子を 0 にリセットする．もし，B-scan パス上の C 素子 s_i の遷移 (0, 1, 0) → (0, 0, 0) に誤りが生じた場合，s_j $(i < j)$ は誤った値をロードし，誤った値を SO まで伝搬する

B-scan パス上の i 番目の C 素子 s_i の (a,b,c)=(0, 1, 0) → (1, 1, 1) の OC 遷移は以下の 3 つの手順を用いテストすることができる．(1) SC1=SC2=11，SI=0 を印加することで B-scan パス上のすべての C 素子を 0 に初期化する．(2) s_i に (a,b)=(1, 1) が印加されるまで，SI=1 を設定しながら，SC1=01，SC2=11 と SC1=11，SC2=01 を繰り返しスキャンシフトを行う．(3) SC1=SC2=11，SI=1 を印加することですべての C 素子を 1 に設定する．s_i に対しては，ロードパターン (1, 1) が印加される前に，(0, 1, 0) の状態に遷移しており，s_1 に対して (a,b,c)=(0, 1, 0) → (1, 1, 1) の遷移が印加される．また，誤りは OF 遷移と同様に SO まで伝搬される．

3.5. 実験結果

本節では提案したスキャン C 素子の可検査性とオーバーヘッドを評価し，文献 [22] で提案されているマルチプレクサベースの手法と比較する．

B-scan パス上の提案したすべてのスキャン C 素子は，3.4.3 節で提案したテス

ト手法を用いることで完全なテストが可能である．一方で，B-scan パス上の既存のスキャン C 素子は上述の方法ではテストすることができない．既存のスキャン C 素子のテストを行う場合は，“scan enable” をオフにしてテストを行う必要があり，スキャンパスを用いずに回路中の C 素子のテストを行うのは現実的ではない．もし可能であったとしても完全なテストを行うのは困難である．

提案したスキャン C 素子のオーバーヘッドを評価するために，複数の種類のスキャン C 素子を標準セルと組み合わせループを用いて実装した．用いた論理合成ツールは Synopsys Design Compiler C-2009.06 であり，class.db と nangate45nm.db の 2 種類のライブラリを用いて論理合成を行った．それぞれのライブラリの面積の単位は NAND ゲート換算でのゲート数，μm^2 である．また，これらのライブラリにはラッチの遅延情報が記されていなかったので，ラッチの遅延を α として扱う．表 3.3 に論理合成結果を示す．ただし，“class.db”，“nangate45nm.db” の下の “Area” はスキャン C 素子の面積，“Delay” は機能パス (A または B から始まり，Z を終点とするパス) の最大遅延を示し，“Original” は DFT なしの C 素子，“Double” はダブルラッチタイプのスキャン C 素子，“Single” はシングルラッチタイプのスキャン C 素子，“Multiplexer” はマルチプレクサベースのスキャン C 素子を示す．“Proposed(mux)” は提案したスキャン C 素子を図 3.4 のように実装し，スキャン制御論理に対してのみ論理最適化を行った．“Proposed(opt)” は提案したスキャン C 素子のスキャン制御論理に対して，機能パスの最大遅延としてマルチプレクサベースの手法で得られた遅延値を与え，スキャン制御部の論理最適化を行った結果を示す．実験結果より，提案したスキャン C 素子は完全なテストが可能であり，オーバーヘッドの面で一番優れる既存の手法と遅延オーバーヘッドを改善，面積オーバーヘッドはほぼ同等で実現できることを示した．

3.6. 結言

今までに提案されている非同期式回路に対する L1L2*スキャン設計手法は，テストパターンとキャプチャパターンの間の依存関係により，完全なテストを保証していなかった．本章では完全なテストに影響を与える依存関係を取り除く 2 部

表 3.3 スキャン C 素子の論理合成結果

	class.db		nangate45nm.db	
	Area	Delay	Area	Delay
Original	4	1.02	2.660	0.08
Double	18	2.40+2α	10.374	0.21+2α
Single	13	2.40+α	7.448	0.21+α
Multiplexer	13	2.40	7.448	0.21
Proposed(mux)	19	2.37	10.906	0.20
Proposed(opt)	15	2.11	7.980	0.15

完全スキャン可検査性を提案し，非同期式回路を 2 部完全スキャン可検査性を満たす回路に設計変更する DFT 手法を提案した．

さらに，C 素子の値保持機能をスキャンパスのメモリとして利用する新しいスキャン C 素子を提案した．提案したスキャン C 素子を用いて構成した 2 部スキャンパス上のすべての C 素子は任意の単一入力変化の遷移を用いたテストが可能である．実験結果において，提案したスキャン C 素子は低い面積，遅延オーバーヘッドで実現可能であることを示した．2 部完全スキャン可検査性と提案したスキャン C 素子を用いることで，非同期式回路を完全にテストすることが可能となる．

今後の課題として，2 部完全スキャン設計を行う際に追加されるラッチ数の削減や，スキャン C 素子のテスト系列を削減することが挙げられる．追加するラッチを削減するためには，S グラフの 2 部化と 2 部完全スキャンパスの構成を同時に行うことが考えられる．また，出力変化遷移 (OC) のテスト系列は各スキャン C 素子に対して生成されるため，各遷移のテスト系列の両立性を考慮することでテスト系列の数を削減できる可能性がある．さらに，本論文では論理故障を対象とした完全なテスト手法を提案したが，同期式回路と同様に，非同期式回路に対しても遅延故障に対するテスト手法が，今後必要になると考えられる．非同期式回路に対する遅延故障の定義，モデル化も含め，非常に困難な問題であるが，大きな研究テーマであると考えられる．

第4章　結論

大規模，高速化するVLSIを効率的に設計するためにはGALS設計が必要不可欠である．GALS設計されたGALSシステムは同期式回路と非同期式回路から成り，それぞれに品質の高いテスト手法が必要となる．本論文ではGALSシステムのテスト品質を向上させるために，同期式回路と非同期式回路のテスト品質を向上させる手法をそれぞれ提案した．

4.1. 同期式回路のテスト品質向上

同期式回路に対するテスト容易化設計手法に完全スキャン設計があるが，過剰テストが発生し，テスト品質が低くなるという問題がある．特にパス遅延故障をテストする際には，多くのパスが通常動作では遷移が伝搬しないフォールスパスであり，歩留まり損失を招きやすい．そこで，フォールスパスを判定することができれば，過剰テストを緩和することができるが，大規模なゲートレベル回路に対してフォールスパス判定を行うことは現実的でない．

また，RTL回路から得られる情報を，論理合成後のゲートレベル回路で利用することができれば，高位の設計情報（RTLや高位合成情報）を用いたテスト手法に有用である．本論文ではRTLとゲートレベルの対応関係を得る手法の１つとして，RTL回路中のパスとゲートレベル回路中のパスのマッピング手法を初めて提案した．これまでに提案されている，多数のゲートレベルパスを１つのRTLパスとして扱う多くの手法はRTLとゲートレベルの間の対応を保証するため，論理合成に制約を置いてきた．提案したパスマッピング手法を用いることで，論理合成の制約を緩和でき，これらの手法の適用範囲を広げることができる．例えば，境界保存論理合成を仮定しているRTL設計情報を用いたフォールスパス判定方

法 [10] も，この仮定を置かないで合成された回路に対しても適用可能となる．実験結果から，複数のベンチマーク回路に対して平均 90.9% の RTL パスと 91.0% のビットスライス RTL パスに対応するゲートレベルパスを得ることができた．これは “単一” のビットスライス RTL 信号線とゲートレベル信号線の機能等価関係を用いた手法で得られる，ほぼ最大のパスマッピング率である．

さらに，RTL フォールスパスの情報を最大限ゲートレベルで利用するために，完全なパスマッピングを実現するマッピング箇所保存論理合成手法を提案した．実験結果により，提案手法は面積，遅延オーバーヘッドを緩和し，フォールスパス情報を RTL からゲートレベルへ伝搬する能力が向上したことを示した．さらに，速度マージンへの影響を調べ，提案手法はその影響を緩和したことを示した．

4.2. 非同期式回路のテスト品質向上

非同期式回路に対するテスト容易化設計手法としては，LSSD のスキャン素子を用いて組合せループを切る手法があるが，面積，遅延オーバーヘッドが大きいという問題がある．そこで，そのオーバーヘッドを軽減するために，L1L2*スキャン設計を用いたテスト手法が提案されているが，非同期式回路に対して完全なテストを保証できないという問題があった．本論文では完全なテストに影響を与える依存関係を取り除く 2 部完全スキャン可検査性を提案し，非同期式回路を 2 部完全スキャン可検査性を満たす回路に設計変更する DFT 手法を提案した．

さらに，C 素子の値保持機能をスキャンパスのメモリそして利用する新しいスキャン C 素子を提案した．提案したスキャン C 素子を用いて構成した 2 部スキャンパス上のすべての C 素子は任意の単一入力変化の遷移を用いたテストが可能である．実験結果において，提案したスキャン C 素子は低い面積，遅延オーバーヘッドで実現可能であることを示した．2 部完全スキャン可検査性と提案したスキャン C 素子を用いることで，非同期式回路を完全にテストすることが可能となり，テスト品質を向上させることができる．

謝辞

本研究の機会を与えた下さるとともに，本研究の全過程を通じて絶えず懇切丁寧な御指導，ご助言をいただきました藤原秀雄教授に心から感謝致します．本研究に際して，有益な御指導を頂きました中島康彦教授に深く感謝致します．本研究にあたり，有益な御指導，御助言を頂きました井上美智子准教授に心から感謝致します．本研究の全過程を通じて，日頃より絶えず貴重な御討論，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教に心から感謝致します．本研究にあたり，有益な御指導，御助言を頂きました米田友和助教に心から感謝致します．本研究に際して，有益な御指導，御助言を頂きました広島市立大学の井上智生教授，吉川祐樹助教に深く感謝致します．本研究に際して，有益な御指導，御助言を頂きました（株）半導体理工学研究センター (STARC) の宮本俊介技監，大西洋一上級研究員,（株）ルネサステクノロジの松島潤氏，シャープ（株）の岡田和久氏，三洋電機（株）の向野守氏に深く感謝致します．最後に，本研究を進めるにあたり，日頃より御協力頂いたコンピュータ設計学講座の諸氏に感謝致します．本研究は一部，半導体理工学研究センター (STARC) との共同研究，及び，科学技術振興機構 (JST) の戦略的創造研究推進事業 (CREST) の研究領域「ディペンダブル VLSI システムの基盤技術」の「フィールド高信頼化のための回路・システム機構」の共同研究，及び，日本学術振興会科学技術研究費補助金・基盤研究 B (課題番号 20300018），及び若手研究 B (課題番号 22700054) の研究助成による．

参考文献

[1] International technology roadmap for semiconductors 2009 edition: Design, 2009.

[2] 当麻喜弘, 南谷崇, 藤原秀雄. フォールトトレラントシステムの構成と設計. 槇書店, 1991.

[3] Henrik Hulgaard, Steven M. Burns, and Gaetano Borriello. Testing asynchronous circuits: a survey. *Integration, the VLSI Journal*, Vol. 19, No. 3, pp. 111–131, 1995.

[4] K.-T. Cheng and H.-C. Chen. Classification and identification of nonrobust untestable path delay faults. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 15, No. 8, pp. 845–853, August 1996.

[5] Seiji Kajihara, Kozo Kinoshita, Irith Pomeranz, and Sudhakar M. Reddy. A method for identifying robust dependent and functionally unsensitizable paths. In *Proceedings of International Conference on VLSI Design*, pp. 82–87, January 1997.

[6] Yun Shao, Sudhakar M. Reddy, Seiji Kajihara, and Irith Pomeranz. An efficient method to identify untestable path delay faults. In *Proceedings of the 10th Asian Test Symposium*, pp. 233–238, 2001.

[7] Angela Krstić, Srimat T. Chakradhar, and Kwang-Ting (Tim) Cheng. Testable path delay fault cover for sequential circuits. In *Proceedings of European Design Automation Conference with EURO-VHDL '96*, pp. 220–226, September 1996.

[8] Ramesh Tekumalla and P.R. Menon. Identifying redundant path delay faults in sequential circuits. In *Proceedings of the Ninth International Conference on VLSI Design*, pp. 406–411, January 1996.

[9] Mehrdad Nourani and Christos A. Papachristou. False path exclusion in delay analysis of RTL structures. *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 10, No. 1, pp. 30–43, February 2002.

[10] Yuki Yoshikawa, Satoshi Ohtake, and Hideo Fujiwara. False path identification using RTL information and its application to over-testing reduction for delay faults. In *Proceedings of the 16th Asian Test Symposium*, pp. 65–68, October 2007.

[11] Thomas Edison Yu, Tomokazu Yoneda, Satoshi Ohtake, and Hideo Fujiwara. Identifying non-robust untestable RTL paths in circuits with multi-cycle paths. In *Proceedings of the 17th Asian Test Symposium*, pp. 125–130, November 2008.

[12] 池田直嗣, 大竹哲史, 井上美智子, 藤原秀雄. 高位合成情報を用いたRTLフォールスパス判定. Technical Report 482, Feb. 2008.

[13] Satoshi Ohtake, Naotsugu Ikeda, Michiko Inoue, and Hideo Fujiwara. A method of unsensitizable path identification using high level design information. In *Conference: International conference on Design & Technology of Integrated Systems in nanoscale era*, Mar. 2010.

[14] Srivaths Ravi, Indradeep Ghosh, Vamsi Boppana, and Niraj K. Jha. Fault-diagnosis-based technique for establishing RTL and gate-level correspondences. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 20, No. 12, pp. 1414–1425, December 2001.

[15] Andreas Kuehlmann and Florian Krohm. Equivalence checking using cuts and heaps. In *Proceedings of the 34th Design Automation Conference*, pp. 263–268, June 1997.

[16] Synopsys, inc. *Formality User Guide*, c-2009.06 edition, June 2009.

[17] Yuki Yoshikawa, Satoshi Ohtake, Tomoo Inoue, and Hideo Fujiwara. A synthesis method to alleviate over-testing of delay faults based on RTL don't care path identification. In *Proceedings of the 27th IEEE VLSI Test Symposium*, pp. 71–76, May 2009.

[18] Yuki Yoshikawa, Satoshi Ohtake, Tomoo Inoue, and Hideo Fujiwara. Fast false path identification based on functional unsensitizability using RTL information. In *Proceedings of Asia and South Pacific Design Automation Conference*, pp. 660–665, January 2009.

[19] Aristides Efthymiou, John Bainbridge, and Douglas A. Edwards. Test pattern generation and partial-scan methodology for an asynchronous SoC interconnect. *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 13, No. 12, pp. 1384–1393, 2005.

[20] Satoshi Ohtake and Kewal K. Saluja. A systematic scan insertion technique for asynchronous on-chip interconnects. In *Digest of papers of Workshop on Low Power Design Impact on Test and Reliability*, May 2008.

[21] Frank te Beest, Ad Peeters, Kees Van Berkel, and Hans Kerkhoff. Synchronous full-scan for asynchronous handshake circuits. *Journal of Electronic Testing*, Vol. 19, No. 4, pp. 397–406, 2003.

[22] Frank te Beest and Ad Peeters. A multiplexor based test method for self-timed circuits. In *Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems*, pp. 166–175, 2005.

[23] Feng Shi and Yiorgos Makris. Testing delay faults in asynchronous handshake circuits. In *Proceedings of 2006 IEEE/ACM international conference on Computer-aided design*, pp. 193–197, 2006.

[24] S. DasGupta, P. Goel, R. G. Walther, and T. W. Williams. A variation of LSSD and its implications on design and test pattern generation in VLSI. In *Proceedings of International Test Conference*, pp. 63–66, 1982.

[25] Miron Abramovici, Melvin A. Breuer, and Arthur D. Friedman. *Digital Systems Testing and Testable Design*. Wiley-IEEE Press, 1994.

[26] Jordi Cortadella, Michael Kishinevsky, Alex Kondratyev, Luciano Lavagno, and Alex Yakovlev. Petrify: A tool for manipulating concurrent specifications and synthesis of asynchronous controllers. *IEICE Transsaction on Information & Systems*, Vol. 80, No. 3, pp. 315–325, 1997.

[27] J. Sparso and S. Furber. *Principles of Asynchronous Circuit Design - A system perspective*. Kluwer Academic Publishers, 2002.

[28] K.M. Fant and S.A. Brandt. NULL convention logic^{TM}: a complete and consistent logic for asynchronous digital circuit synthesis. In *Proceedings of International Conference on Application Specific Systems, Architectures and Processors*, pp. 261–273, 1996.