NAIST-IS-MT0951052

修士論文

通常動作を考慮した遅延故障テスト生成に関する研究

児玉 秀憲

2011年2月3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
修士 (工学) 授与の要件として提出した修士論文である。

児玉 秀憲

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）

通常動作を考慮した遅延故障テスト生成に関する研究*

児玉 秀憲

内容梗概

VLSI 設計では，テストを容易にするためにスキャン設計と呼ばれるテスト容易化設計が行われる．スキャン設計では外部から任意の値をフリップフロップに設定できるようにすることで，生成されたテストパターンを組合せ回路に印加することができる．しかし，通常動作では起こりえない状態遷移を起こすことができるようになるため，通常動作では活性化されない故障を検出するパターンオーバーテストが発生する．また，スキャンテストはテスト時には通常動作時と異なる電力を消費する．テスト時の消費電力が通常動作より大きくなると，電圧降下を引き起こすことがあり，電力オーバーテストと呼ばれる正常チップを不良品と誤判定してしまう歩留り劣化の問題が生じる．消費電力が小さくなると，テスト時に通常の厳しい電力状況が再現できず，電力アンダーテストが発生し，不良品の見逃してしまう．この問題を解決するために，テスト時に回路が通常動作に近い振る舞いをする遅延故障テスト生成法を提案する．提案法ではテスト生成において，テストパターンに無効状態が現れないようにするためのテスト生成制約回路を作成し，これを用いて通常動作に現れる状態遷移に近いテストを生成する．これにより，通常動作時と同等の消費電力でのテストを行い，歩留り劣化の問題を解決する．提案法ではパターンオーバーテストの削減，電力オーバー/アンダーテストを緩和することができた．

* 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 修士論文，NAIST-IS-MT0951052, 2011 年 2 月 3 日.

キーワード

遅延故障テスト生成，通常動作テスト，Launch-off-capture

Studies on Functional Operation-Aware Delay Fault Test Generation*

Hidenori Kodama

Abstract

In designing VLSI, design for testability called scan design is used to facilitate test. In scan design, it is possible to apply test pattern to combinational circuit by being able to set any value on flip-frops from the outside. However, pattern that cannot be set to flip-flops during functional operation can be shifted-in and detect faults whitch is not sensitized during functional operation. this is called pattern over testing. In scan testing, power which is different from functional operation is consumed during test operation. If power is larger during test operation than functional operation, IR-Drop may be happened and test may judge good chips as defective products wrongly. This is called power over testing. On the other hand, if power is smaller during test operation than functional operation, it is impossible for test to induce severe power situation and it may cause test escape. This is called power under testing. In order to solve this problem, in this paper, a method of functional operation-aware delay fault test generation is proposed. In the proposed method, test generation constraint circuit is created and attached to the test generation model so as not to embed invarid states in test patterns, and using it generate test whitch has transition state during functuinal operation. Test patterns generated by the method can consume power which is in the range of power consumed in normal operation and then the method can alleviate yield

* Master's Thesis, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT0951052, February 3, 2011.

loss and test escape. Proposed method reduces pattern over testing and relieve power over and under testing.

Keywords:

Design for testability, functional operation-aware test, launch-off-capture

目 次

1. 序論 **1**

1.1 研究背景 1

1.2 本論文の目的 2

1.3 本論文の構成 3

2. 諸定義 **5**

2.1 RTL 回路 9

2.2 論理合成 11

2.3 遅延故障 13

2.4 テスト容易化設計 15

2.4.1 順序回路 15

2.4.2 スキャン設計 16

2.4.3 遅延故障テスト生成法 17

2.4.4 Launch-off-capture 18

2.4.5 パターンオーバーテスト 18

2.5 テスト生成 19

2.6 テストパターン解析 20

2.6.1 テストパターンの品質 21

2.7 消費電力 21

2.8 スキャンテストの消費電力 23

2.8.1 シフト時電力 23

2.8.2 キャプチャ時電力 24

2.8.3 電力オーバーテスト 24

2.8.4 電力アンダーテスト 24

2.9 キャプチャ電力の削減 25

3. 提案手法 **27**

3.1 既存手法 27

3.2 テスト生成制約回路による遅延故障テスト法の提案 30
3.2.1 RTL からの展開 31
3.3 有効状態と無効状態 33
3.4 テスト生成制約回路 34
3.5 テストパターン生成制約法 37

4. 実験 **40**
4.1 実験環境 40
4.2 実験方法 41
4.2.1 消費電力解析 41
4.3 実験結果 43
4.4 考察 52

5. 結論 **53**

謝辞 **55**

参考文献 **56**

図 目 次

1 VLSI の設計フロー 5
2 ゲートレベル設計 13
3 スキャン設計 16
4 Launch-off-capture のタイミングチャート 18
5 テスト生成のフロー 19
6 CMOS 回路 22
7 フルスキャン回路 25
8 制約を付加した 2 時刻展開モデル ($N + 2$ 時刻展開モデル) 29
9 X-filling 手法 29
10 データパスとコントローラーの関係 31
11 コントローラー部分に記述されている状態 32
12 paulin 回路の状態信号割り当て 33
13 paulin 回路の無効状態から作成したテスト生成制約回路 35
14 paulin 回路のテスト生成制約回路 36
15 テスト生成制約回路を作成する HDL 記述 36
16 テスト生成制約回路を合成したスキャン回路図 37
17 順序回路のシフトレジスタ 38
18 テスト生成制約回路を合成したスキャン回路図 39
19 テストパターンの消費電力解析フロー 42
20 s1_reset 回路の各テスト生成 Launch 部分の電力解析結果 48
21 s1_reset 回路の各テスト生成 capture 部分の電力解析結果 48
22 pma_reset 回路の各テスト生成 Launch 部分の電力解析結果 49
23 pma_reset 回路の各テスト生成 capture 部分の電力解析結果 49
24 s832_reset 回路の各テスト生成 Launch 部分の電力解析結果 50
25 s832_reset 回路の各テスト生成 capture 部分の電力解析結果 50
26 dk16_reset 回路の各テスト生成 Launch 部分の電力解析結果 51
27 dk16_reset 回路の各テスト生成 capture 部分の電力解析結果 51

表 目 次

1 paulin 回路の状態割り当て表 34
2 テスト生成制約回路の真理値表 35
3 実験環境 40
4 RTL ベンチマーク回路特性表 41
5 s1_reset 回路の各テスト生成結果 44
6 s1_reset 回路の各テスト生成の消費電力解析結果 44
7 pma_reset 回路の各テスト生成結果 45
8 pma_reset 回路の各テスト生成の消費電力解析結果 45
9 s832_reset 回路の各テスト生成結果 46
10 s832_reset 回路の各テスト生成の消費電力解析結果 46
11 dk16_reset 回路の各テスト生成結果 47
12 dk16_reset 回路の各テスト生成の消費電力解析結果 47

1. 序論

VLSI(Very Large Scale Integration:超大規模集積回路) は膨大な数の電子素子を 1 つのチップに組み込むことで高機能を実現したものの総称であり，携帯電話，自動車，家電製品等の多くの電気製品に搭載されている．半導体集積化技術の発展は目覚ましく，集積度の向上によって，従来基盤を組み合わせることで実現されていたシステム全体を 1 つの集積回路上に統合する技術である SoC(System on Chip) を実現できるようになり，また，2015 年には 100 億個のトランジスタが 1 チップに構築されるとまで言われている．半導体集積度の向上に起因するこれら IC(Integrated Circuit:集積回路) の発展によって，我々の生活が豊かになってきたことは言うまでもない．しかし，集積回路技術の向上によって，1 チップに搭載される素子数やそれに伴う電気的制約が増加するだけでなく，高速化，低消費電力化，不良率の減少を考慮する必要があり，VLSI 設計はますます煩雑化している．このため，これらの要求の実現に対して優れた VLSI 自動設計技術が注目を集め，盛んに研究が行われている一方，VLSI を製造し工場出荷する過程において，不良品を減らすための故障診断性技術の向上，信頼性の高い品質を提供する先端的な技術も生み出されている．本章では，本論文における研究の背景とともに，その有意性について述べる．

1.1 研究背景

IC は，要求された複雑な機能を果たすために，抵抗器，インダクタンス，キャパシタンス，半導体等の電子部品を組み合わせることで様々な機能を達成する集積回路であり，我々の身の回りに存在する様々な電化製品になくてはならない重要なものとなっている．情報処理や信号処理等を行う上で欠かせない主要要素であり，情報通信基盤設備から個人ユースの PC，モバイル機器，ディジタル家電に至る幅広い電子機器に用いられている．我々が身の回りの製品に高度な機能を求めれば求めるほど，より高度な半導体製品を必要とし，その度に半導体の微細化技術は向上されてきた．LSI の集積度が向上するにつれ，VLSI チップに搭載される素子数が増加することで，素子間の電気的制約や低消費電力など，様々な目

的を同時に最適化しなければならなくなり，より高品質なレイアウトやそれに伴う故障検出テストが求められるようになってきた．

一般的にVLSIの設計は，まず要求仕様が与えられ，その後要求分析，システム設計，アーキテクチャ設計，高位合成，論理合成，レイアウト設計というそれぞれの工程を経て数億個程度の回路素子を搭載し，要求を満足するようなチップを実現する．これらの要求を実現するためにはVLSIの高度な設計技術が必要であり，そのため，自動設計技術が盛んに研究されているが，トランジスタの微細化により複雑化し先端化したVLSIは，チップ面積や低消費電力等に加え，モジュール間の電気的制約や，テスト容易化設計への対応等の高度な設計が要求され，その設計は複雑化する一方である．

このように，今なお進展しているVLSIシステムの大規模化，複雑化により，VLSI設計開発は長期化する傾向にある．しかしながら，市場における商品のライフサイクルは短くなる一方であり，そのため，高品質なVLSI自動設計が強く求められている．この要求を満たす信頼性の高い高品質なVLSIを製品として製造するためには，テストおよび故障解析の重要性が大変増してきている．従来は，回路内の論理値が0あるいは1に固定される縮退故障が多く使われ，診断とともに実際に検出された不良は縮退故障で特定できるものが多かった．しかし，近年の半導体の微細化の進展に伴い，従来では顕在化しなかった製品の欠陥が不良品の原因として問題視されており [1] ，特に遅延時間が変化するような遅延故障の増加が顕著である．この遅延故障は現在の半導体製造において大変悩ましい問題であり，どれ程の短期間に高品質な製品を製造し出荷できるかというこの目標に立ちはだかっている．大規模回路に対して微小な遅延不良の影響もシステム動作に影響を及ぼすようになってきているため，遅延故障に対する故障診断の要求はますます高いものになっている．

1.2 本論文の目的

近年，LSI設計は抽象度の高いレベルのアルゴリズムで記述され，自動合成ツールを用いてゲートレベルや標準セルレベルの設計を生成を行う．既存のほとんどの合成システムでは，動作速度，面積，消費電力などに挙げられる回路の性能の

最適性を考慮する．また，回路の故障の有無を検証するテスト容易性については，ゲートレベル回路に対してスキャン設計によって確保するのが一般的である．しかし，スキャン設計にはテストデータ量が大きい，スキャンシフト時の消費電力が大きいといった様々な解決しなければならない問題があり，スキャン設計に関する研究ではこれらの問題を改善する多数のアプローチがなされている．

高性能 LSI に対してタイミング不良に対する高品質テストを生成する技術の確立がますます重要になっており，遅延故障に対するテスト生成が提案されている．回路中には活性化されないパスが多数含まれており，そのパス上の遅延故障はテスト不可能となっている．特に順序回路の遅延故障テスト生成ではそれらの故障がテスト不可能であることを判定するために大きな時間がかかり，高品質なテストを得ることが困難となっている．

これまで，スキャン設計などのテスト容易化設計をすることで，問題の故障をテスト可能にすることでテスト生成に費やす時間の削除を行ってきた．しかし，活性化されていないパス (フォールスパス) 上の遷移で発生する遅延故障をテスト可能にすることで LSI は過剰テストとされ，歩留り低下といった深刻な問題を引き起こすパス遅延故障，論理ゲートの入出力を構成する信号線の遅延が増大する遷移遅延故障などに代表されるような遅延故障が大きな問題となっている．

このようなゲートレベルにおける故障診断，スキャン設計の改善ではなく，回路の合成段階や上流設計に対するスキャン設計とは別のテスト容易化設計法も提案されている．本研究では，通常動作を考慮した遅延故障テスト生成法を提案する．具体的には，RTL の設計情報を用いて有効状態のみを持つテストパターンを生成するために，テスト生成アルゴリズムに制約を与えるのテストパターンに制約を与える回路を作成し通常動作を考慮したテストパターン生成を行うことで，テスト時の消費電力，計算時間，信号値の遷移頻度の削減，テスト生成に制約を与えることの有用性を検証する．

1.3 本論文の構成

本論文は以下に示す章により構成される．

第 1 章では，本研究における背景と研究目的を述べる．

第 2 章では，緒定義として LSI 設計のフローを概説し，その中でも本論文で取り上げる遅延故障テスト生成，通常動作テスト，Launch-off-capture について詳しく述べる．さらに，通常動作と時テスト時の消費電力問題について述べる．

第 3 章では，本論文の提案手法を紹介し，テスト生成制約回路の設計法，テストパターン生成法について述べる．

第 4 章では，提案手法に基づく実験を行い，提案法の検証，評価を行う．

最後に第 5 章で，本論文の結論を述べる．

2. 諸定義

図 1 に VLSI の基本的な設計フローを示す．実際の回路設計においては，問題が生じた場合，多くの後戻り工程が生じたり，設計仕様の評価解析・機能検証などが必要となるため，実際は図 1 のフローよりもさらに複雑となる．本章では，それぞれの工程における詳細を説明し，本論文の内容であるテスト容易化設計，故障モデル，テスト生成法，テストの消費電力問題について詳しく述べる．

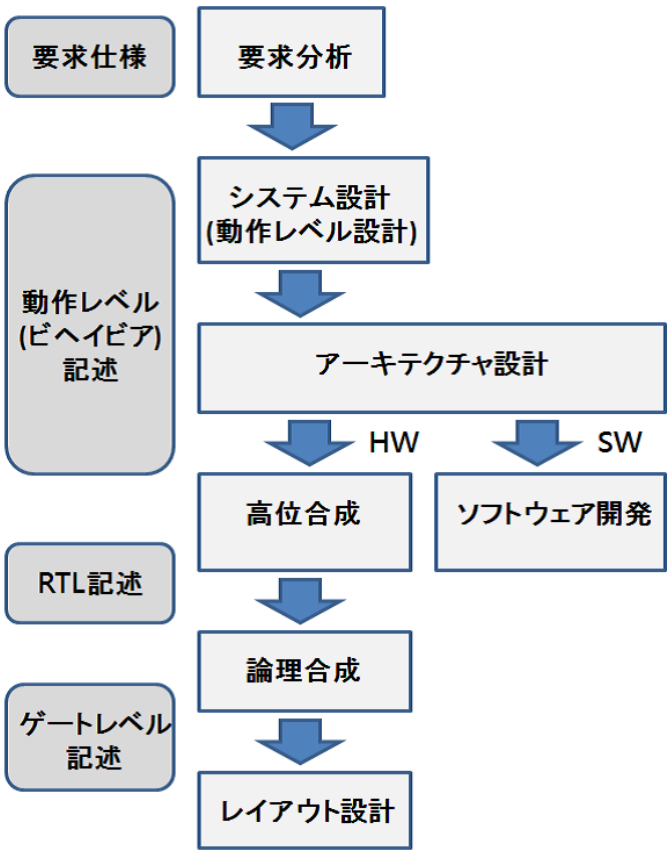

図 1　VLSI の設計フロー

VLSI の設計は，設計制約を考慮した動作仕様を決定する「要求分析」，システムの実現方式を決定する「システム設計」，ハードウェアとソフトウェアを分割し，それぞれをどのようなアーキテクチャ(構造) で実現するかということを検討する「アーキテクチャ設計」，ハードウェア化する機能のビヘイビアを RTL(Register Transfer Level：レジスタ転送レベル) 記述に変換する「高位合成」，RTL 記述をゲートレベル記述へ変換する「論理合成」，モジュールの配置配線等を考慮する「レイアウト設計」からなる．また，要求分析，システム設計，アーキテクチャ設計，高位合成までの高位設計と，論理合成，レイアウト設計の下位設計に大別される．高位設計は実際の回路図やソフトウェアを作成するための具体的な図を作成し，下位設計は高位設計で作成された設計図をもとに機能を具現化するための工程である．本節ではそれぞれの設計工程の内容について述べる．

要求分析

開発者は，ユーザの要求を市場等から把握 (要求分析という) し，その上にコスト，信頼性，メンテナンス性等を考慮して製品の仕様を決定しなければならない．これらの要求は，要求定義書と呼ばれるものにまとめられる．要求定義書には次の項目が含まれる．

- システムの目的.
- 解決すべき課題.
- システムと外部とのインターフェース.
- システムの機能.
- システムの稼働環境.
- 開発上の制約.

これらの項目を考慮することによって，要求定義書で動作周波数，処理速度，消費電力等のチップの性能を決定する．また，設計者は要求定義書から具体的な設計をする必要があるため，専門的に理解しやすい表現方法で要求仕様書を作成

する．ここで，要求仕様書とは，要求定義書に書かれた要求を実現するために，仕様という専門的な形式で記述されたもののことをいう．要求仕様書は，自然言語で書かれる場合が多い．なお，ニーズはハードウェア，ソフトウェアを包含した記述であるが，設計の途中段階ではハードウェアとソフトウェアに分割して設計される．

システム設計

チップに対する仕様が決まると，システム設計工程において，仕様を満足するように，実現すべき機能とそれらの関係について記述したシステム仕様を作成する．このシステム仕様は，チップ上で稼動する組み込みソフトウェアも合わせたシステム全体を対象とし，機能を表現する手法としてビヘイビアモデルを用いてブロック図や文章で記述される．システム仕様もまたシステム仕様書として文書にされる．また，システム仕様がシステムレベル設計言語で記述されている場合，プログラムリストがそのままシステム仕様書となる．

一般的に，製品の欠陥は下位で発見されるほど後戻り工程の規模が大きくなり，時間とコストがかかってしまうので，要求仕様をシステムに実現するための設計の初期段階である本工程において十分な検証と，下位における設計を容易にするための設計が重要となる．

アーキテクチャ設計

この工程では，システム仕様書に記述された内容を実現するためのシステム構成を決め，要求された機能をハードウェアで実現するか，ソフトウェアで実現するかというハードウェア/ソフトウェア分割を行う．近年のシステム LSI のような，複雑な機能を多く持ち，プロセッサやメモリが組み込まれているような VLSI においては，すべての機能をハードウェアで実現することは現実的ではなく，一部の機能は組み込みプロセッサとメモリ上でソフトウェアとして実現される．アーキテクチャ設計では，ハードウェア/ソフトウェア分割を行った後，それぞれの仕様をビヘイビアレベルで記述する．ここでは，ハードウェア/ソフトウェア分割を効率よく進めるために，協調設計 (co-design) と呼ばれる手法が用いられ，ハードウェア設計とソフトウェア設計は協調しながら行われる．システム仕様がシス

テムレベル設計言語で記述されているときは，システムがソフトウェアで表現されているため，どの部分をハードウェアで実現するかを決める作業がハードウェア/ソフトウェア分割に相当する．

1. プラットホーム選択

   システム LSI では，アプリケーションプログラムを実行するための組み込みプロセッサを内蔵するのが一般的であるため，どのような種類のプロセッサやバスを採用するのか，画像処理や外部とのデータ授受にはどのような規格のインターフェースを採用するのか，などといったシステムを構築するための環境を選択をする．これらの組み合わせは，プラットフォームと呼ばれる基本となるシステムの組み合わせがいくつか存在する．このプラットフォームを選ぶことをプラットフォーム選択といい，システム仕様に照らしあわせて最適と考えれえるプラットフォームを選択する．プラットフォーム選択により開発環境が決定されると，次に，想定したシステムがどの程度の演算量，データ伝送量であるか，また入出力ポートはいくつ必要か，などといったシステムの特性を見積もる工程であるプロファイリング (profiling) を行う．このプロファイリングの結果により，後のハードウェア/ソフトウェア分割の最適化やプラットフォームの変更等が行われる．また，システム LSI が必要とする機能は，映像処理や音声処理など他のシステム LSI と共用できる部分が多く存在し，これらは機能ブロックとして再利用される．このような機能ブロックのことを IP(Intellectual Property) と呼び，マイクロプロセッサ，各種コントローラーなども IP 化されている．この IP は流用する設計段階によって記述が異なるため，様々なモデルのものがあるが，アーキテクチャ設計においてはビヘイビアモデルや RTL モデルのものが用いられる．

2. ハードウェア/ソフトウェア分割

   ハードウェア/ソフトウェアの分割を最適化する場合，性能を最大化するか，コストを最小化するかなどといった指標を明確にしなければならない．これらはトレードオフの関係になり，同時に最適化できない場合が多い．従って，設計開発前には，製品のおおよそのコンセプトを決めておかな

ければならない．具体的な作業としては，ハードウェア/ソフトウェア協調設計ツール (コデザインツール:co-design tool) と呼ばれる設計用ソフトウェアに，ハードウェア化する機能ブロックとソフトウェア化する機能ブロックのそれぞれのビヘイビアを指定することでシステムの動作シミュレーションを行い，システムの要求を満足するような分割方法を決定する．また，協調設計ツールは分割されたハードウェアとソフトウェアのインターフェースを自動生成する機能もあり，インターフェースもまたハードウェア化されるものとソフトウェア化（ソフトウェア化したインターフェースはデバイスドライバと呼ばれる）されるものがある．

2.1 RTL 回路

高位合成 (動作合成) とは，アーキテクチャ設計によってハードウェア化する部分のビヘイビアを，HDL(Hardware Description Language:ハードウェア記述言語) により記述された RTL に変換し，ハードウェア仕様を計算機により自動的に作成することをいう．なお，これらの処理は高位合成ツールと呼ばれるものにより自動化される．この工程では，ハードウェアにおける処理を，データ処理回路と制御回路に分けて設計し，また，クロックレベルのタイミング設計も行われる．以下に高位合成の手順を述べる．

1. ビヘイビアの変換・最適化

   ビヘイビアをコントロールデータフローグラフ (CDFG:Control Data Flow Graph) で表す．これは，データパスのデータ処理機能を表すデータの処理順序を表現したデータフローグラフ (DFG) と，制御回路の機能を表す順序制御を表現したコントロールフローグラフ (CFG) を同時に表したものである．CDFG で表されたビヘイビアの動作内容は，結合則や交換則などの演算法則を用いることで演算数を削減し最適化を行う．

2. スケジューリング

   この工程では，高性能化，コストなどの設計要件を考慮し，使用する演算器の個数を決定した後，これらを最も有効的に活用するために，演算のタイ

ミングやレジスタへのデータ退避等を設計する．このスケジューリングには，処理に必要なデータがそろったタイミングで演算を実行する ASAP(As Soon As Possible) スケジューリング，最も遅いタイミングで実行する ALAP(As Late As Possible) スケジューリング，ASAP スケジューリングに演算順で優先度をつけるリストスケジューリング等がある．

3. 割り当て

   CDFG に含まれるデータ処理機能とその制御機能に対し，演算器，レジスタ等を割り当て，RTL の記述を生成する．データパスへのハードウェアの割り当てでは，データ処理に必要な演算，記憶，データ転送機能に対してそれぞれ演算器，メモリ，パスを割り当てる．制御回路はデータ処理の各要素の動作を制御するが，スケジューリングが完了した段階で状態遷移図として制御動作が表現され，これを実現する有限ステートマシンを RTL で表す．

次に，RTL を構成するデータ処理回路 (データパス部分) と制御回路 (コントローラー部分) の設計について詳しく説明する．仕様設計から出力された仕様書をもとに，LSI 内部をいくつかの大きなブロックに分割する．このブロック分割を何度も繰り返すことで，ブロックが組合せ回路とレジスタの接続により構成されるところまで回路を詳細化する．ブロックを組合せ回路とレジスタの接続で表現することにより，1 クロックごとの回路の動作を細かく記述することができる．

データパスの設計

データパスとは外部からデータを入力し，そのデータと内部に記憶したデータを使用して演算を行い，演算結果を外部に出力したり内部に記憶する回路である．まず最初にブロックの機能である入出力関係を明確にすることで，ブロックを構成する部品となる回路を明らかにする．1 つのブロックが加算，減算といった算術演算だけではなく，論理和や論理積などの論理演算を行う場合は ALU (Arithmetic Logic Unit:演算論理演算器) と呼ばれる演算器が必要となる．そして演算器を使用してさらに別の演算を行う場合は，結果を記憶するためのレジスタが必要となる．また，データの入力元

や結果の出力先が複数ある場合は選択のためのデコーダやマルチプレクサが必要である．演算器，レジスタ，デコーダ，マルチプレクサを RTL 回路の部品とみなし，RTL 回路内で接続される．

このようにデータパスでは，外部からの入力データ信号や内部の複数レジスタの中から 1 つを選択してデータを入力し，演算を行い，外部への出力データ信号や内部の複数のレジスタの中から 1 つを選択して演算結果を出力する．

コントローラーの設計

データパスを思いどおりに動作させるには，外部から順序正しく適切なタイミングで制御信号を与えなければならない．このようにデータパスに制御信号を与え，データの流れや演算の種類，順序をを制御する機能を司るのがコントローラーである．

一般に状態遷移図で記述された動作は，次の状態の決定と現在の状態の保持と各状態の制御信号の決定に分けられる．これらの動作をハードウエアで実現させるためには，現在の状態と入力信号から次の状態を決定する組合せ回路と，現在の状態を保持するレジスタと，現在の状態と入力信号から制御信号の値を決定する組合せ回路が必要となる．このような次状態決定部，状態保持部，制御信号決定部から構成される回路はステートマシンと呼ばれる．コントローラーではこのステートマシンによりデータパスの機能を制御する．

2.2 論理合成

論理合成では，高位合成で得られた RTL 記述から，論理素子 (AND/OR/NOT など) と記憶素子 (フリップフロップ) で構成される論理回路を生成する．下記に論理合成の手順を示す．

1. RTL の論理式への変換

   RTL 記述に含まれる if 文，for 文，case 文，算術演算等を構文解析し，ルー

ルとして定義されている回路変換を行う．この工程では，算術演算器数の削減処理としてリソースシェアリングと呼ばれる処理も行われている．

2. 論理の最適化

　まず，論理レベルで最適化が行われ，その後，論理式から，論理ゲートで構成されるゲートレベルの回路を生成する．このとき，論理式の簡略化や信号伝達経路にふくまれるゲート数を最小化する．また，ゲートレベルの論理ゲート，基本セルは，テクノロジマッピングと呼ばれる操作で割り当てられる．

論理合成ではRTL設計から出力されたRTL記述を入力として論理回路図を得る．RTL回路を構成する加算器やALUなどの組合せ回路やレジスタなどの順序回路を，ANDゲートやORゲート，フリップフロップなどの基本回路を接続してゲートレベルによる回路表現で構成する．

現在ではRTL設計の段階でHDLを用いてRTL回路を記述し，HDLシミュレータで検証を行う．その後，論理合成ツールを用いて自動的にゲートレベルに展開していくことで，セルにマッピングすることが可能となっている．このような設計法はトップダウン設計と呼ばれ，従来の設計法と比べて設計効率が大幅に上昇している．RTL仕様からレイアウト設計までのフローを図2に示す．

ゲートレベルのセルはトランジスタで構成される．回路設計でセルの論理的な機能を実現するように電気的特性を考慮しながらトランジスタの接続関係とゲートのサイズを決定する．次の工程であるレイアウト設計において，セルをチップ上に配置し，セル間の配線経路を決定する．このとき，セルの物理的な形状・寸法とコンタクト位置などの情報を持ったライブラリデータを使用して配置配線を行い，レイアウトする．

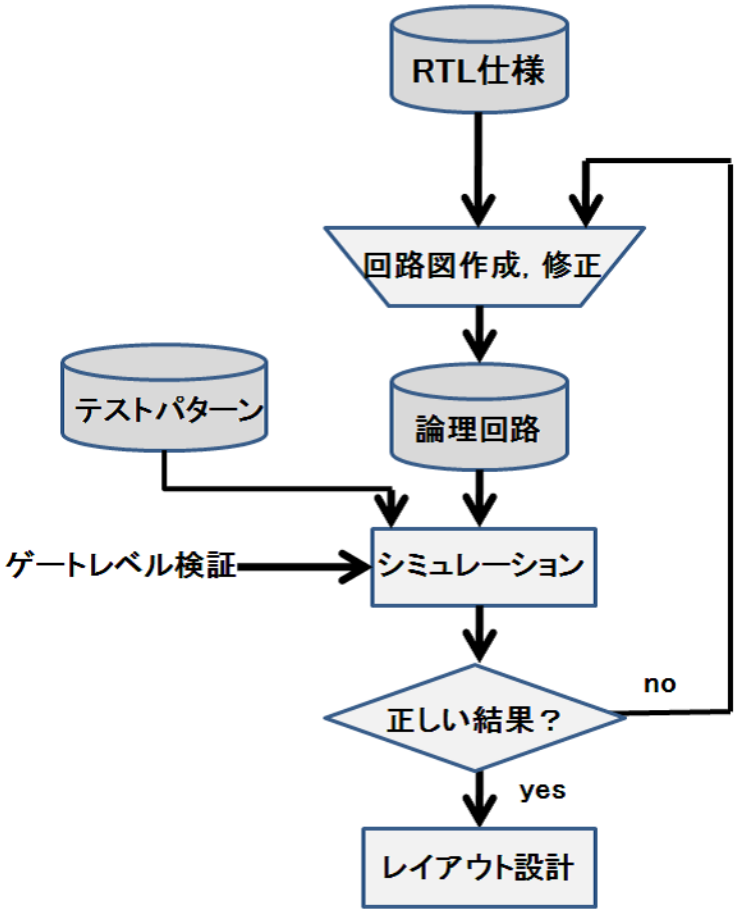

図 2　ゲートレベル設計

2.3　遅延故障

遅延故障とは，順序回路を構成する素子や配線の物理的欠陥により遅延が増加することで，回路が正しく機能しなくなり，誤動作が生じる故障である [2]．フリップフロップの出力で生じた信号値変化が組合せ回路で伝搬して次のクロックでフリップフロップに取り込まれる際，決められた範囲を越してしまう遅延時間が原因で誤動作が発生する．遅延故障は各故障部分に信号の立ち上がり (0 から 1 への変化) の遅延と立ち上がり (1 から 0 への変化) の遅延の 2 種類に分類される．遅延故障は信号値の変化に影響されるため，遅延故障を検出するためには変

化前の信号値を設定するテストパターンと変化後の値を確かめる計2つのテストパターンを用いて連続して印加しなければならない．この遅延故障テスト生成法を2パターンテストという．

現在，遅延故障モデルとして大きく遷移遅延故障モデルとパス遅延故障モデルの2つがある．遷移遅延故障はゲートピンに比較的大きな遅延故障モデルを仮定し，その活性化パス，および伝播パスにかかわらず検出可能な故障モデルであり，パス遅延故障はある特定のパスが遅延故障を起こすと仮定し，そのパスを活性化させるためのパターンを必要とする．通常，パス遅延故障はタイミングを解析することによりクリティカルパスを抽出し，パス遅延テストのターゲットとする．

遷移遅延故障

論理ゲートの入出力を構成する1つの信号線の遅延が増大する故障を遷移遅延故障 (トランジション故障) という．ある信号線の立ち上がり遷移故障を検出するには，信号線の値を0に設定するパターンを印加した後，連続して信号線で信号値の変化が生じるよう1に設定し，信号値変化が生じたことを出力で確認できるパターンを印加する．遷移遅延故障では，回路内で仮定される故障の数が信号線数の2倍 (各信号線の立ち上がり故障と立下り故障) ある．そのため回路内の全ての故障を検出することが可能である．また，遅延故障のテストパターンを生成する ATPG や故障検出率を計算する故障シミュレーション方法は縮退故障を対象にする場合と同様に比較的に容易である．

パス遅延故障

パス遅延故障とはフリップフロップ間を結ぶ特定の信号伝搬経路の遅延が増大する故障である．信号が伝搬する際に累積する遅延をモデル化しており，遅延要素が回路内に小さくかつ分散しているケースに有効である．とあるパス遅延故障がテストできれば，そのパス上の縮退故障や遷移遅延故障などの他の故障もテスト可能である．しかし，回路のパス数はゲート数に対して増大する場合があり，パス遅延故障テスト生成では，全故障を対象とするテストとして現実的に不可能となっている．そのため，パス遅延故障を対象としたテストパターンだけで高いテスト品質を達成することは

容易ではない．

遷移遅延故障は，遅延テスト行う際の対象故障モデルとしてよく用いられる遅延故障モデルであるため，本論文ではこの遷移遅延故障を取り扱う．この故障のテストに用いられる 2 パターンテスト生成法については，後の節で詳しく述べる．

2.4 テスト容易化設計

回路を正しく設計しても製造時に欠陥が発生した場合，回路は正常に動作しない．これを防ぐためには設計した回路が正しく動作することを確認するためのテストが必要である．これは回路の製造上での欠陥が無いかどうかを検査するためのテストである．回路が大規模化，複雑化するにつれて，回路の設計だけではなくテストの設計もまた困難なものとなっている．特に，順序回路はフリップフロップを含むことからテスト生成が困難である．そのため，順序回路のテストに対しては，テスト生成の容易な回路に設計変更するテスト容易化設計が行われる．

2.4.1 順序回路

ゲート回路で構成し，入力の確定で所定の値が出力される回路が組合せ回路なのに対し，順序回路はゲート回路と記憶回路で構成し，入力と記憶回路の値から出力と回路の次の状態が決定する．つまり，順序回路の時刻 t の入力と状態がその時刻 t の出力を決める．デジタル回路の大部分は順序回路であり，記憶回路にはフリップフロップを用いる．順序回路では，一定周期の時刻を刻むタイミング信号をクロックと呼び，クロックごとに入力と状態と出力が変化する．組合せ回路に対しては高速アルゴリズムが研究開発され実用化しているのに対し，順序回路は回路規模が増大するにつれて，テスト系列の生成が困難になり，高い検出率のテストパターンを生成することができない．そこで，スキャン設計を用いることで，テスト生成時に順序回路を組合せ回路として扱う．フルスキャン順序回路では，すべてのフリップフロップをスキャンフリップフロップに置き換えることにより，シフトとキャプチャの 2 つのモードで動作する．順序回路のテストでは，スキャンテストは広く用いられており，大規模回路では必要不可欠となっている．

2.4.2 スキャン設計

スキャン設計は順序回路に対する最も一般的なテスト容易化設計である．スキャン設計を図 3 に示す．

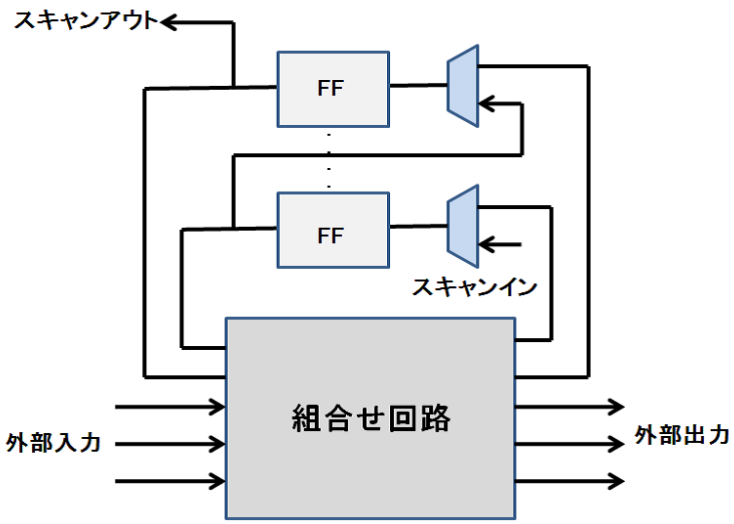

図 3 スキャン設計

スキャン設計とはテストモード時に回路内のレジスタをシフトレジスタとして構成し，回路の外部からレジスタの値を設定，観測できるようにすることである．スキャン設計によってテスト容易化の指標である回路の可制御性と可観測性が向上する．スキャン設計により順序回路を組合せ回路としてみなすことができ，スキャンテストの自動生成が容易になる．スキャンテストでは，回路をテストモードにしてレジスタに値を設定し (スキャンイン)，回路を通常動作モードにしてクロックを入れて動作させ，レジスタに前段の組合せ回路の値を取り込み，回路をテストモードにしてその値を外部に取り出す (スキャンアウト) 動作が行われる．

テスト回路の合成ではレジスタをスキャンレジスタに変更し，スキャンチェーンが挿入される．これは論理合成ツールで自動的に行うことができる．テスト合成では，設計した回路の欠陥を発見するためのテストパターンを生成することである．これは ATPG ツールを用いて自動的に行うことができる．

2.4.3 遅延故障テスト生成法

スキャン設計がなされた順序回路では，テストパターンの印加とその応答の観測を，スキャンシフト動作を通じて行う．遅延故障を検出するためには 2 パターンテストを行うが，高い故障の検出率を得るためには，スキャン設計だけではなくテストパターンの印加方法，スキャンフリップフロップの構成を工夫しなければならない．これは 1 パターン目と 2 パターン目が連続して実動作速度で印加することができないからである．1 パターン目の印加と 2 パターン目の出力の観測にスキャン動作は有効だが，2 パターン目の印加には原則としてスキャン動作を使用することはできない．スキャン回路に対して 2 パターンテストを印加する方法としてシフト (Launch-off-shift:LOS) とキャプチャ(Launch-off-capture:LOC) の 2 つのモードで行う．

Launch-off-shift 方式

LoS 方式では，スキャンイン動作の最後の 2 シフトクロックをそれぞれ 1 パターン目，2 パターン目と設定する．その後，スキャンイン直後に実速度で 1 回キャプチャする．スキャンインを行う際に，最後のクロックとキャプチャクロックの間隔が実速度になるよう要求されるが，その短い間にスキャンシフト動作から通常動作へスキャンイネーブル信号を切り換えなければならない．2 パターン目に任意のパターンを印加するが，通常動作では使わない状態遷移に基づく動作でテストを行うため，オーバーテストが発生しやすい．

本論文では 2 パターンテスト生成を取り扱うため，LoS 方式よりも Launch-off-capture 方式を用いて遅延故障テスト生成を行うこととする．これは，スキャンシフトにおいて 1 ビットシフトする LoS 方式よりも組合せ回路を通してキャプチャを行う LoC 方式の方が設計制約が少ないため一般的だからである．

2.4.4 Launch-off-capture

LoC 方式では，シフト動作にて 1 パターン目をスキャンイン (入力) からスキャンチェーンに設定する．する．このテストパターンを組合せ回路に印加し，通常動作でシステムクロックにより出力応答 (キャプチャ) を 2 パターン目として設定し，キャプチャを行う．スキャンインした後に，システムクロックにより 2 回キャプチャを行う．図 4 にクロック信号とスキャンイネーブル信号のタイミングチャートを示す．

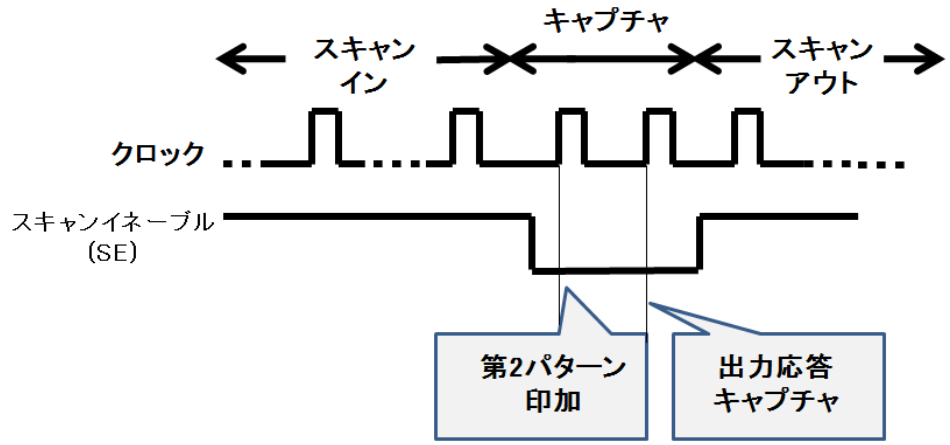

図 4 Launch-off-capture のタイミングチャート

LoC 方式の特徴として， 2 パターン目が通常動作と同じなので設計時の制約が少ないことが挙げられる．また， LoS 方式よりもスキャンシフト動作から通常動作への切換に時間的な余裕があるため，標準のスキャン設計で十分実現可能となる．

2.4.5 パターンオーバーテスト

このように，スキャン設計はテスト生成が容易になる半面，通常動作では活性化されない故障 (冗長故障) を検出するパターンオーバーテストが発生してしまう．スキャン設計では，第 1 パターンに任意の値を印加することができる．これ

は，通常では設定できないパターンを用いることができるため，検出しなくて良い故障までもを検出してしまい，テストパターンによるオーバーテストが起きてしまう．こうして歩留り損失を生じてしまう．

2.5 テスト生成

テストパターン生成では，回路の故障を検出するために入力パターンの系列を回路に印加してその出力系列を観測する．テスト生成を行った後，回路に故障が存在するかどうかを判定することを故障検出という．回路に故障が存在することがわかったとき，それがどのような故障であるかどうかを調べることを故障診断という．本論文では遅延遷移故障を検出するものとする．

次に，テスト生成のフローを図 5 に示す．

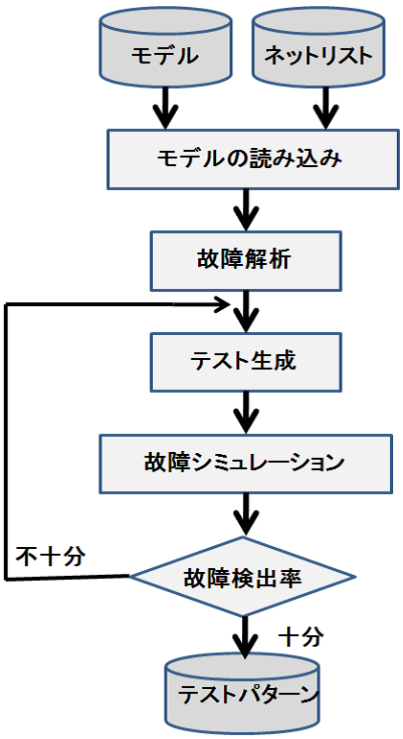

図 5 テスト生成のフロー

テスト生成はコンピュータを利用して様々なVLSIを効率よく設計する技術であるCAD(Computer Aided Design)の一環として捉えられている．ネットリストにはスキャンチェーンの入出力ポート，回路をテストモードに設定するためのポート，スキャンチェーンのシフトを可能にするポートなどのスキャン設計された回路に関する情報をATPGツールに与えることで，要件に合わせたネットリストを作成し処理を行う．ネットリストとライブラリモデルを読み込むことでゲートレベルのATPG回路モデルを構築し，自動でテストパターン生成を実行する．故障シミュレーションでは，故障がある回路と正常な回路に対してテストを加えた場合の回路動作のシミュレーションを行い，与えられたテストパターンがどの故障を検出するかを調べる．

2.6 テストパターン解析

生成されたテストパターンと故障シミュレーションの結果から故障クラスが作成される．以下に示すのはATPGツールを用いた際の主な故障クラスの種類である．

DT(detected)故障

検出故障と言い，シミュレーションによる検出した故障の数が表される．

UD(undetectable)故障

冗長故障と言い，いかなる方法を用いても観測ができないためテストができず，故障を検出できないことを表す．

AU(ATPG untestable)故障

ATPGツールを用いて検出できなかった故障のことを指す．ATPGツールを用いた際にテストできない故障を含む場合があり，これはUD故障とは示されず，機能テストのような他の手法を使用することでテストできる場合がある．

ND(not detected)故障

未検出(アボート)故障と言い，テストで検出できなかった故障である．回

路が複雑すぎると，ATPG アルゴリズムで上手く解釈できないことがあるため，ND 故障が多数検出される場合がある．これは ATPG の試行レベルを上げることによって，テスト検出率の向上が可能となる．

2.6.1 テストパターンの品質

生成されたテストパターン集合で, 想定した故障のうちどれだけの故障が検出されるかを示す指標を故障検出率という．また，テスト生成アルゴリズムが想定した故障のうち，検出または冗長と判定した比率を故障検出効率というテストパターンの品質は以下の式で得ることができる．

$$\text{故障検出率} = \frac{\text{検出故障数}}{\text{故障総数}}$$

$$\text{故障検出効率} = \frac{\text{検出故障数} + \text{冗長と判明した故数}}{\text{故障総数}}$$

一般的に，冗長故障が存在すれば故障検出率は 100%にならないが，冗長故障数を識別さえすればたとえ冗長故障が存在してもテスト生成アルゴリズムが優秀であれば故障検出率は 100%になる．

2.7 消費電力

順序回路のテストにおいて，フルスキャン設計と組合せ回路の用の ATPG に基づくスキャンテストについて述べてきたが，この節ではそのテスト時の消費電力問題について詳しく述べる．一般的に，テスト時の消費電力は通常動作時の消費電力よりも大きい [4][5]．これは，テスト時にスキャンチェーンを用いることにより，通常動作では起こりえない値の設定や動作が発生するからである．通常動作時では複数の機能ブロックが相互的に依存しているため，高速動作でのみスイッチングが発生していたが，テスト動作時ではシフトクロックが働くためシフト動作とキャプチャ動作における速度が同じにならないため，機能ブロックの相互的な依存関係が無視される．これがテスト時の消費電力が高くなる原因である．

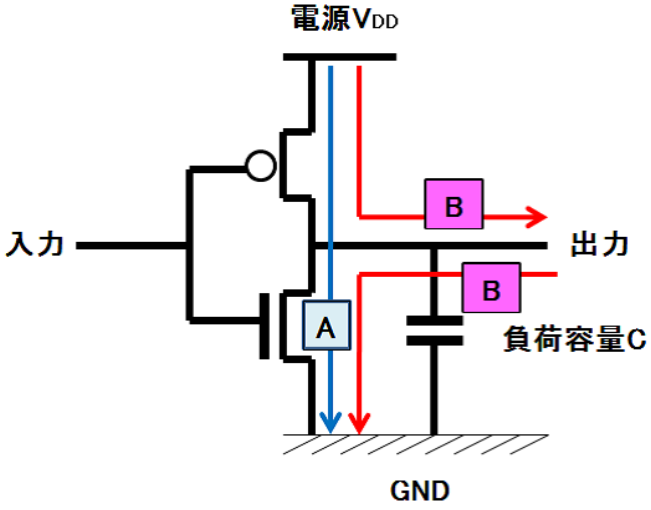

図 6　CMOS 回路

CMOS(Complementary Metal Oxide Semiconductor) 回路を図 6 に示し，消費電力について説明する．

CMOS 回路は消費電力が小さいことが利点であるが，クロックなど内部の論理回路の論理が変動する場合，内外に付随する容量により電流が変化する仕組みになっている．これは内外の容量にある電荷が充放電されるのが原因である．そのため，動作速度にほぼ比例して消費電流が増えていくので，高速で回路動作の実行が繰り返された場合，平均して見ると消費電流が増大するので大量の電力を消費して発熱を伴う．

CMOS 回路の消費電力はスイッチング電力，貫通電流による電力，リーク電流による電力の 3 つに分類され，特にスイッチング電力と貫通電流による電力をダイナミック電力，リーク電流による電力をスタティック電力と呼ぶ．

図 6 より，CMOS 回路の消費電力のうち，B は信号線のスイッチングによって発生する消費電力であり，CMOS 回路全体の消費電力の 70%以上を占める要素である．A は，セル内の貫通電流による消費電力であり，LSI 全体の消費電力の 10〜30%を占める．本論文では，B のスイッチングを行うダイナミック電力の消費電力問題を取り扱う．このダイナミック電力の CMOS 回路における消費電力の

計算式を次に示す.

$$P = P_t \times C \times {V_{DD}}^2 \times f$$

上記の式より，P は消費電力，P_t は信号遷移確率，C は負荷容量，f は周波数，V_{DD} は電源電圧である．

ダイナミック電力は，CMOS 回路の出力が 0 から 1，1 から 0 へとスイッチングしたときの負荷容量の充放電によって電力を消費する．すなわち，0 と 1 が切り替わる度に電流が流れる．図 6 が示すインバータの出力が 0 から 1 に遷移するとき，正電源 V_{DD} から p-MOS を通って負荷容量 C へ電流が流れる．すると p-MOS が負荷抵抗となるので，これを通過して電流が流れる際に p-MOS で発熱が起こると同時に電力が消費される．インバータの出力が 1 から 0 に遷移する場合も同様に行われる．

2.8 スキャンテストの消費電力

通常動作には，組合せ回路からの入力を受けてフリップフロップに値が入る動作があるが，これは回路の一部にしかスイッチングが発生しない高速動作である．しかし，スキャンチェーンを用いることにより，通常動作では起こりえない値の設定や動作が発生する．次に，スキャンイネーブルからの信号で通常動作からテスト動作に切り替えた場合，動作として回路全体に通常動作では見られなかった大きなスイッチングが発生することになる．スキャンテストのスイッチング電力には，シフト時電力とキャプチャ時電力が存在する [6].

2.8.1 シフト時電力

テスト動作におけるシフト動作では，スキャンフリップフロップをスキャン入力，出力ピンから直接制御し観測するためにスキャンチェインを動作させる．シフト時の消費電力は，テストベクトルがスキャン回路のスキャンチェーンに印加されるときに発生する．このシフト時の消費電力が増大すると，スキャンフリップフロップ間でデータ転送 (遷移) が正常に起こらず予期したテストパターンが印

加できなくなる恐れがあり，回路が発熱し破壊されてしまう．このように消費電力が大きくなると永久的なダメージを生んでしまい，信頼性が低下する．

2.8.2 キャプチャ時電力

テスト動作におけるキャプチャ動作では，テストパターンを組合せ回路に印加し，その出力応答を観測する．キャプチャ時の消費電力は，テストベクトルによる組合せ回路部分の出力応答がフリップフロップへ取り込まれる際に，そのフリップフロップが保持している値と取り込まれる応答の値が異なった場合に発生する．値が遷移するフリップフロップの数が多くなるほど，多くの電力が消費される．

2.8.3 電力オーバーテスト

前節で述べたキャプチャ時の消費電力が増大すると，瞬間的な IR ドロップ (電圧降下) が発生する．過剰な消費電力が流れて発生する IR ドロップは，通常動作時だけでなくテスト動作にも起こる．特にテスト動作の IR ドロップは，フリップフロップの誤動作やゲート間の遷移の遅延の増加などを引き起こす．通常動作時の IR ドロップはある程度の範囲内で許容されるよう設計されているが，テスト動作時には許容量をオーバーする IR ドロップが発生する場合がある．その主な要因として，スキャンシフト時の過剰なスイッチングやテスト実行時に実動作速度以上のシステムクロックが設定されるなどがある．その結果，テスト時に正常な素子をまで故障と判断してしまい，誤って回路を不良品と判定してしまう電力オーバーテストを招き，歩留りが低下する原因となる．

2.8.4 電力アンダーテスト

通常動作で起こりえる IR ドロップやノイズなどが原因で，電力がテスト時に励起できなくなる場合もある．スキャンテストが実行された際，テスト動作時に必要十分な電力が使用されるため，通常動作時では許容量を超えない程度の電力しか使用できない厳しい電力状況にある．そのため，十分な消費電力が使用され

ないことが原因で，本来ならば検出されるべき故障がテストされないという電力アンダーテストが起こってしまい，不良品の見逃しの原因となる．

2.9 キャプチャ電力の削減

次にキャプチャ時の消費電力の様子を具体的な回路図と共に図 7 で示す．図 7 は一般的な順序回路のフルスキャンテストである．テストベクトルを v，v の外部入力 PI と擬似外部入力 PPI のビットをそれぞれ $(v : PI)$，$(v : PPI)$，組合せ回路部分の論理回路を F，v に対する応答を $F(v)$ とする．$F(v)$ の外部出力 PO と擬似外部出力 PPO のビットをそれぞれ $F((v) : PO)$，$(F(v) : PPO)$ とする．

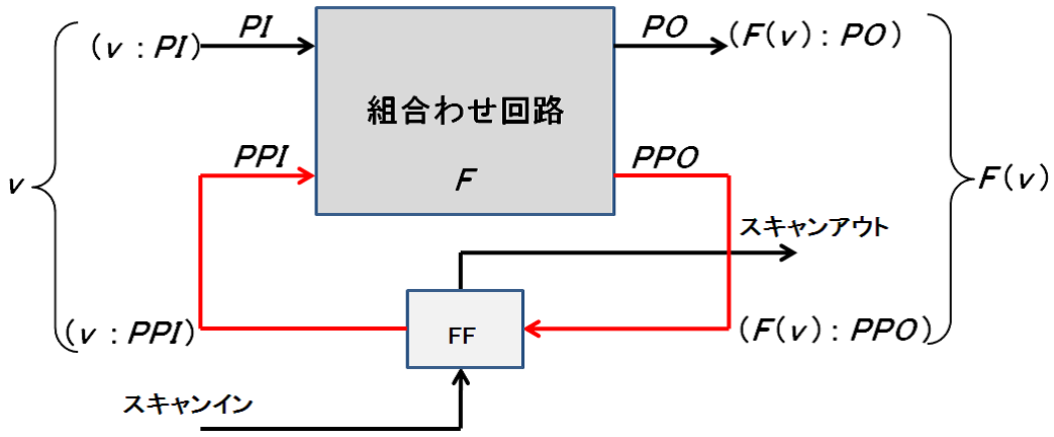

図 7　フルスキャン回路

スキャンフリップフロップで $(v : PPI)$ のあるビット n がそれに対応する $(F(v) : PPO)$ のビット n と反対の論理値を取ったとすると，スキャンテストのキャプチャ時にフリップフロップの出力で値の遷移が起きる．これがキャプチャ動作によるスイッチングである．v に対する一度に対するスイッチングの個数を $CT(v)$ とする．v のすべてのビットに値が割り当てられた場合は次のように表わすことができる．

$$CT(v) = |(v : PPI) \oplus (F(v) : PPO)|$$

$CT(v)$ は，テストベクトル v によって引き起こされる回路のスイッチングの数から影響を受ける．この $CT(v)$ が大きくなると，$(F(v) : PPO)$ が取り込まれた際に組合せ回路部分で何度もスイッチングが起こり，IR ドロップが生じる可能性がある．このことが原因でゲートで遅延が起こりキャプチャ時のタイミング違反が起きてしまう．すなわち，不正な値をキャプチャしてしまい誤ったパターンを出力することになる．

このキャプチャ時の誤動作による歩留り低下を減らすには，テストベクトル v のキャプチャ遷移数 $CT(v)$ を減らす必要がある．

ただし，v と $F(v)$ が通常動作で起こるものであれば，パターンオーバーテスト，電力オーバー/アンダーテストにもならない．本論文では，このスイッチングに着目して遷移数 $CT(v)$ を削除できるような手法を提案し，通常動作に近づけたテスト生成法を検証する．

3. 提案手法

前章では諸定義として，順序回路のフルスキャン設計の仕組みや遷移遅延故障検出に有効である Launch-off-capture 方式，スキャン設計の応用としてテスト動作の消費電力に関する問題点について詳しく述べた．この章では，諸定義からの問題提起を元に本研究の目標である通常動作を考慮した遅延故障テスト生成法の提案をする．

問題提起として，スキャンチェーンを用いることにより通常動作では起こりえない値の設定や動作が発生し，通常動作では活性化されない故障を検出するパターンオーバーテスト，回路の通常動作とは異なる電力が消費されて発生する電力オーバー/アンダーテストといった回路の損傷および故障の見逃し，歩留り劣化が挙げられる．これらの問題を防ぐためには，高品質なテストを保証する手法として通常動作に近い動作でテストを行う必要がある．この章では，過去に提案された低消費電力を目指したスキャン設計法を紹介し，その欠点を補う方法として本研究の提案手法について述べる．その提案手法とは，テスト生成に制約を与えたテストパターンを印加してその応答出力を得て，遷移遅延故障のテストパターンの期待値を出力する方法であり，この結果と順序回路のテストパターンを比較して通常動作に近づいているのかどうか，シミュレーションを行う．

3.1 既存手法

シフト時の消費電力の削減を目的とした手法は数多く提案されており，以下の手法がある [7].

テストスケジューリング

IP コアベース設計において，消費電力を考慮しながら同時にテストするコアを選抜することで電力量をある一定の上限値以下に制御する

テストベクトル操作

低消費電力 ATPG，静的圧縮，テストベクトル圧縮

回路操作

遷移セルブロック，クロックゲーティング

スキャンチェーン操作

スキャンチェーンの順序変更やスキャンチェーン分割

このように，シフト時の消費電力の削減では，シフト時の遷移を考慮する重み付き遷移の手法が一般的である．

それに対し，キャプチャ時の消費電力の削減は容易ではない．まず，以下の 2 つの方法を説明する．

1. N フレーム展開モデルをテスト生成制約回路として使用する手法 [8]
   宮田が提案したこの手法は，印加するパターンが通常動作時のものに近くなるように，テスト生成の際にフリップフロップに設定する値に制約を与えることで通常動作時に近い電力消費でテストを行うことを目標としたものである．

   既存の組合せ回路用のテスト生成アルゴリズムを用いて 2 時刻展開モデルをとし，遅延故障テストパターンを生成する．Launch-off-capture の動作では，第 1 パターンの擬似外部入力に任意のパターンを印加可能であるため，テスト生成アルゴリズムは通常動作では現れない状態を生成する可能性がある．この手法では，第 1 パターンの擬似入力部の値をできるだけ通常動作時に用いる状態になるようにするため，2 時刻展開モデルのフレーム 1 の擬似入力部に制約回路を付加し，テスト生成アルゴリズムの探索空間を制約する方法として組合せ回路を N フレーム展開し，これを制約回路として接続する．この回路を $N+2$ 時刻展開モデルと呼ぶ．$N+2$ 時刻展開モデルを図 8 に示す．スイッチングを減らすためのテストパターンを制約する方法として，1 時刻目と 2 時刻目で外部入力値に遷移を発生させないようにする．時刻展開したい回路を用いてその外部入力の 1 時刻目と 2 時刻目で連結するため，回路規模が複雑で大きくなると制約回路も大きくなる．そのため，この手法の欠点として，テスト生成時間が膨大になってしまう．

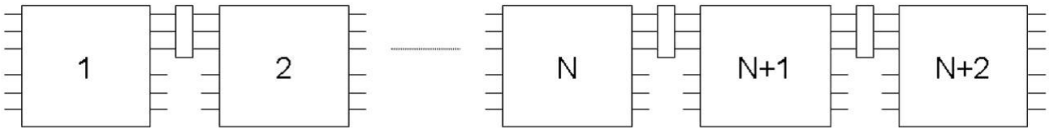

図 8　制約を付加した 2 時刻展開モデル ($N+2$ 時刻展開モデル)

2. X-filling 手法 [9]
   この手法はテストキューブの中の X に 0 と 1 を適切に割り当てることで，得られるテストベクトルを制御する．X はドントケア (don't cares) といい，その値を 0 にしても 1 にしても故障検出率に影響のないビットである．図 9 に X-filling 手法の手順を示す．

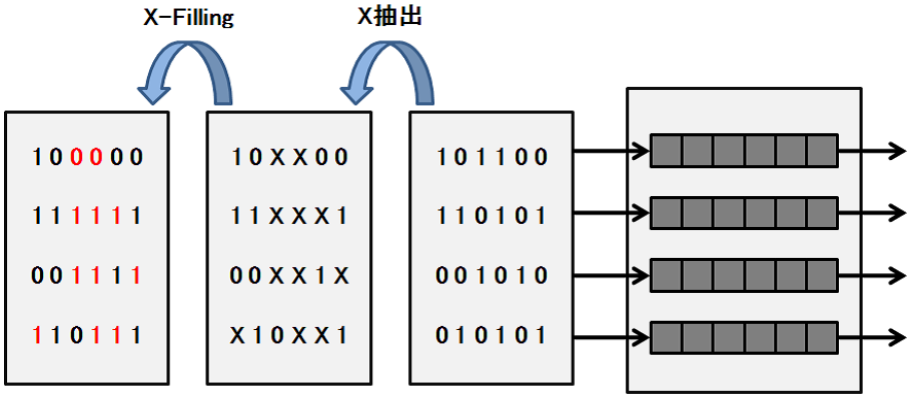

図 9　X-filling 手法

出力されたテストパターンに対して X-Filling(Minimum Transition Fill) を行う．X の値を最少になるように修正し正当化操作を行うことで，テストパターンとそのテスト応答のハミング距離が小さいテストパターンを生成する．これにより，キャプチャ時の低消費電力化を達成することができる．テストパターンの並び替えの際に，複数のテストパターンを印加する必要が有る．テストパターンの印加順序に依存するため最適な順序を求めるの

に時間がかかる．

2 時刻展開モデルは通常動作時に近いテストパターンの生成が行えると期待するフレーム数 N を大きくすると，より元の順序回路の動作に近づくと考えられていたが，テスト生成時間が膨大になってしまう問題点があった．また，X-filling 手法では，ゲートレベルでテストパターンの無効状態を解析するため，探索時間が大きくなってしまう問題点がある．

本論文では，キャプチャ時のスイッチングを考慮したテスト生成アルゴリズムを提案し，通常動作に近づけた新しい Launch-off-capture テスト生成法について検証する．

3.2 テスト生成制約回路による遅延故障テスト法の提案

提案手法では，テスト生成において，テストパターンに無効状態が現れないようにするためのテスト生成制約回路を作成し，これを用いてテスト生成アルゴリズムを制約する．すなわち，通常動作に現れる状態遷移に近いテストを生成する．

1. RTL 情報
   前述したように，今までのテスト生成に制約を与えて通常動作に近いパターンを求める手法では，テスト生成時間がかかりすぎてしまう問題点があった．しかし，提案手法では RTL 情報を用いて無効状態を高速に抽出することができる．まず，データパス部分とコントローラー部分に分ける．

2. 有効状態と無効状態の抽出
   RTL から得たコントローラー部分を解析し，内部状態に着目する．抽出した内部状態は有向グラフで表現した際にお互いに到達可能な状態であるため，この到達可能状態を含むテストセットを作成し，回路から多くの遷移遅延故障を検出する．通常動作時に到達可能な状態が有効状態，それ以外が無効状態となるため，RTL のコントローラー部分から有効/無効状態が高速で求められる．

3. テストパターンを制約する組合せ回路の作成
   有効状態，無効状態の情報から組合せ回路を設計する．この組合せ回路をテスト生成時に制約回路として用いて，Launch-off-capture を対象にした通常動作に近いテストパターン生成を行う．

3.2.1 RTL からの展開

RTL 設計された回路の内のデータパスとコントローラーの関係 [10] を図 10 に示す．

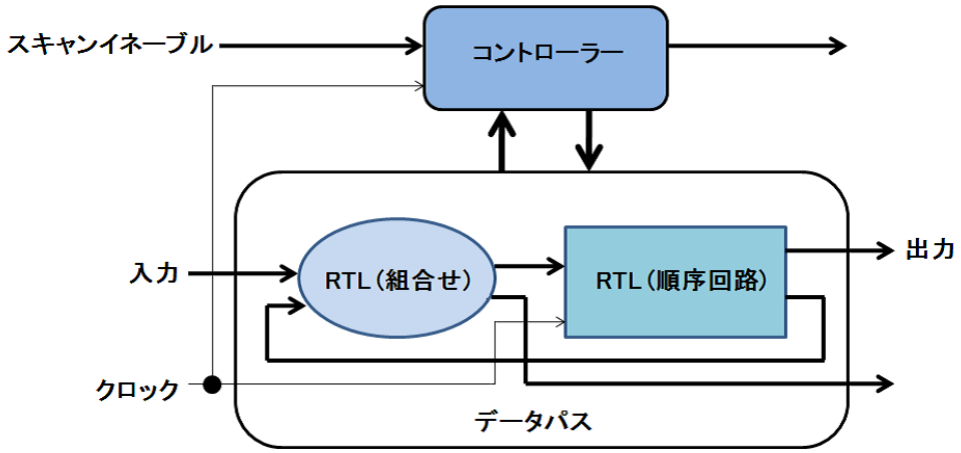

図 10　データパスとコントローラーの関係

データパスには自由な値が入るため，テストパターンを制約するためにはコントローラーにある内部状態を見て，テストパターンに使用されるテストセットを作成することができる有効状態を高速で抽出する．すなわち，コントローラーから得られる有効状態を用いて制約回路を作成する．

VHDL で設計された RTL 回路からコントローラ部の有効状態を抽出する方法を例を挙げて説明する．

RTL の構造は entity と architecture の 2 つの部分から構成される．entity 部分ではモデルの入出力であるインターフェースが記述されており，architecture 部分ではモデルの機能と回路の構造が記述されている．そして，次のコンポーネント宣言で entity のインターフェース情報であるコンポーネントのポートと信号を接続する記述が現れる．図 11 に RTL ベンチマークの paulin 回路のコントローラー部分の状態を示す.

```
architecture rtl of controller is
      type state_type is (s0, s1, s2, s3, s4, s5);
          signal ps, ns : state_type;
```

図 11　コントローラー部分に記述されている状態

図 11 から各状態 S_0〜S_5 が見て取れる．この state_type の宣言によりコントローラーの状態が記され，信号割当てが行われる．

3.3 有効状態と無効状態

内部状態から割り出すことができる状態は，初期状態から到達可能な状態と到達不可能な状態に分けられる [11]．paulin 回路の内部状態は前述の通り，{$s0$, $s1$, $s2$, $s3$, $s4$, $s5$} の 6 状態を確認することができる．$\log_2 6 \cong 3$ より paulin は 3bit 回路なので全ての状態は 8 状態 {$s0$〜$s7$} であり，残りの $s6$ と $s7$ は到達不可能状態 (無効状態) になる．また，論理合成の際に行われた状態割り当て情報を得ることにより，無効状態を知ることができる．図 12 に得られた paulin 回路の状態信号割り当てを示す.

```
Encoding Bit Length: 3
State Vector: { ps_reg[2] ps_reg[1] ps_reg[0] }
State Encodings and Order:
s0          : 000
s1          : 001
s2          : 010
s3          : 011
s4          : 100
s5          : 101
```

図 12　paulin 回路の状態信号割り当て

Encoding Bit Length でスキャン回路のフリップフロップの集まりであるレジスタのビット幅，State Vector で該当するスキャンチェーンの状態レジスタ部分，State Encodings and Order で有効状態に割り当てられるバイナリコードが示される．状態信号割り当ての法則としては，最短ビット幅のコードを順で状態遷移が行われている．

3.4 テスト生成制約回路

次に，テストパターンに有効状態のみを使用し，それを用いてテスト生成アルゴリズムに制約を加えるテスト生成制約回路を作成する．表 1 に paulin 回路の状態割り当て表を示す.

表 1　paulin 回路の状態割り当て表

状態	バイナリコード (信号値)	状態判定
s_0	000	有効状態
s_1	001	有効状態
s_2	010	有効状態
s_3	011	有効状態
s_4	100	有効状態
s_5	101	有効状態
s_6	110	無効状態
s_7	111	無効状態

表 1 より，テストに使用される有効状態は $\{s0,\ s1,\ s2,\ s3,\ s4,\ s5\}$ で信号値が $\{000,\ 001,\ 010,\ 011,\ 100,\ 101\}$ に対し，テストに使用されない無効状態は $\{s0,\ s7\}$ で信号値が $\{110,\ 111\}$ であることがわかる．通常動作に近付けるためには，Launch-off-capture の第 1 パターンから無効状態を取り除くことができるように，テスト生成に制約を与える必要がある．テスト生成に制約を与えるために，無効状態からテスト生成制約回路を作成する．このテスト生成制約回路はゲートレベルの回路である.

作成したテスト生成制約回路は，スキャン設計された回路に合成して新たなスキャン設計回路とし，テスト生成を行う．これにより，テスト生成に制約を与えることが可能である．このテスト生成制約回路の特性として，有効状態が入力された場合にはその状態と同じ値を出力する．無効状態を通した場合にはある特定の有効状態になって出力される．これは後にテストパターンを変換する必要があ

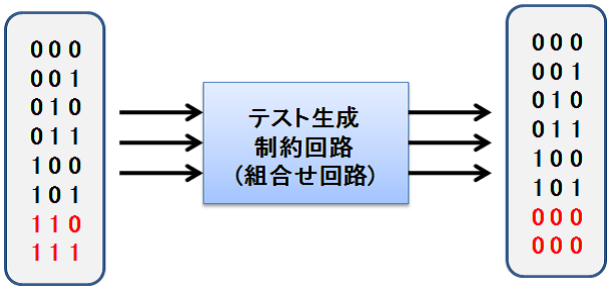

図 13　paulin 回路の無効状態から作成したテスト生成制約回路

るためである．図 13 に paulin 回路のテスト生成制約回路によるパターン変換を示す．

図 13 のテスト生成制約回路の入力が無効状態である 110 と 111 の場合，有効状態を出力するが，態割り当て法則よりバイナリデータの最初の有効状態である 000 を出力するものとする．

- **真理値表から作成する設計法**

表 2 から真理値表を作成する．その真理値表を表 2 に示す．

表 2　テスト生成制約回路の真理値表

入力 ABC	出力 A’B’C’
000	000
001	001
010	010
011	011
100	100
101	101
110	000
111	000

表 2 の各出力 A'B'C' からカルノー図 14 を作成する．各出力値 A'B'C' に対応する組合せ回路を統合し，テスト生成制約回路が完成する．

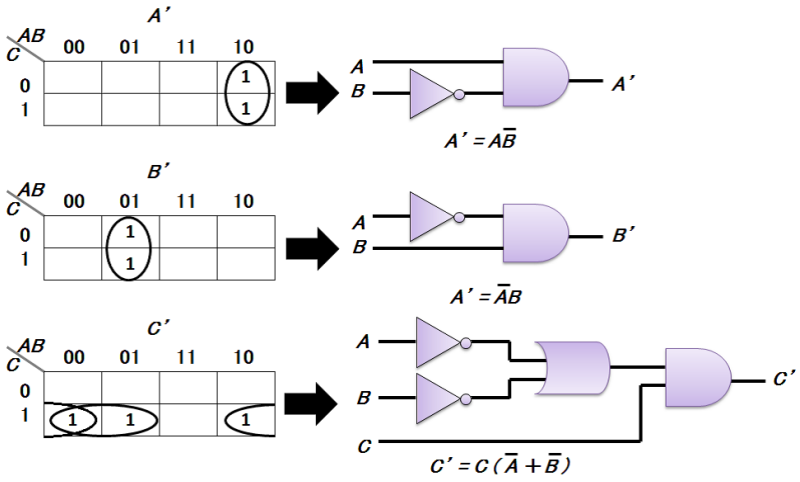

図 14　paulin 回路のテスト生成制約回路

- HDL **記述で作成する設計法**

HDL 記述 [12][13][14] で作成する場合，図 15 に示すようにプロセスの主要部分は if 文で構成されている．if 文の各分岐において，inport の有効状態値に対応して，outport に対する割当てが行われている．この場合，inport に 000〜101 が入るとそのまま inport の入力値を割当て，else 文で inport に無効状態値が入ると 000 を出力する．

```
if inport <= "101" then
      outport <= inport;
        else
      outport <= "000";
```

図 15　テスト生成制約回路を作成する HDL 記述

3.5 テストパターン生成制約法

作成したテスト生成制約回路を用いてテスト生成に制約を与え，制約された新しいテストパターンを求める．テスト生成制約回路の入力先をスキャンチェーンの状態レジスタ部分の出力に，テスト生成制約回路の出力先を組合せ回路の擬似外部入力先に接続することで，図 16 に示すようなテスト生成の制約が完成する．すると，テスト生成制約回路を通るパターンは，合成した回路の制約によってすべて有効状態のみを含むテストパターンとなり，キャプチャ動作で組合せ回路に印加される．スキャンフリップフロップにより任意の値ではないパターンを印加することで，通常動作に近付けた動作で電力を消費する遷移遅延故障テスト生成を行う．

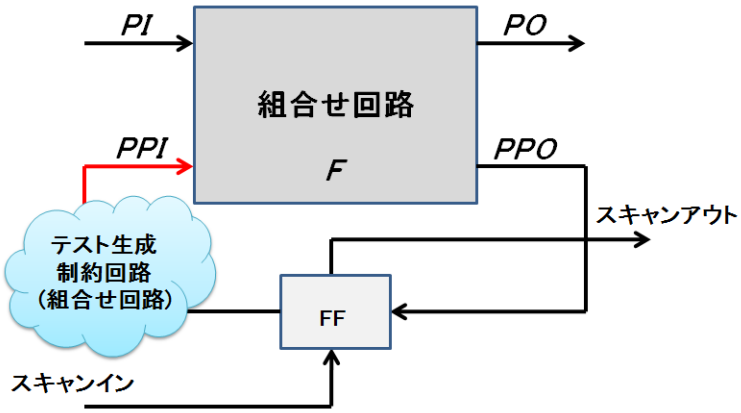

図 16　テスト生成制約回路を合成したスキャン回路図

シフトレジスタには n ビット幅分の D フリップフロップ型の状態レジスタ接続されており，入力データが D から入ることにより，次状態 Q と $\bar{Q}$ の値が出力される．順序回路のシフトレジスタの構成を図 17 に示す．

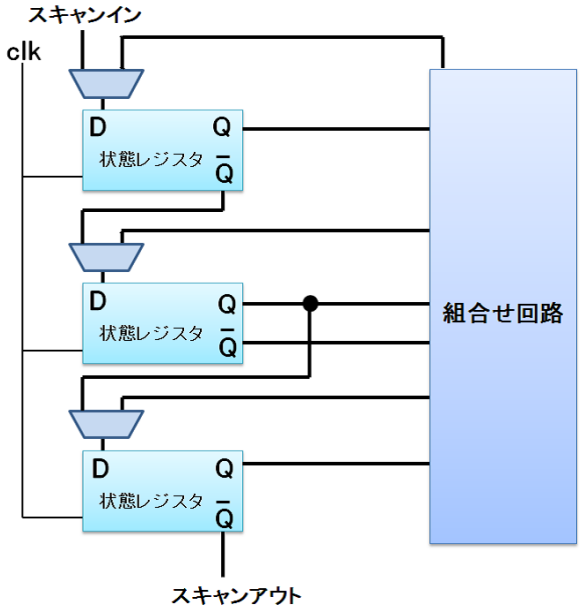

図 17　順序回路のシフトレジスタ

また，順序回路をフルスキャン設計にしたテスト用回路にテスト生成制約回路を合成した様子を図 18 に示す.

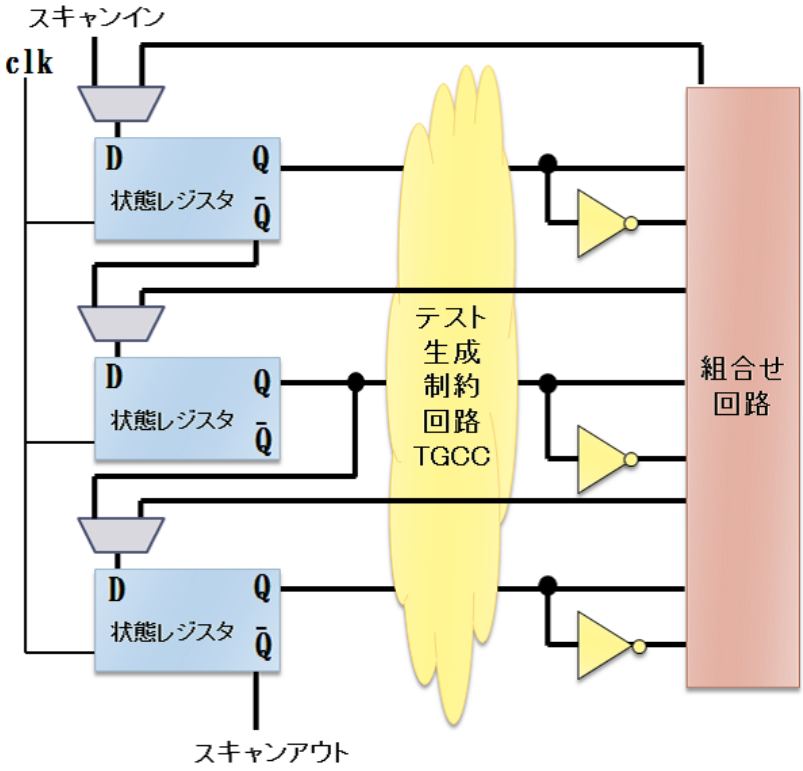

図 18　テスト生成制約回路を合成したスキャン回路図

状態レジスタの出力 Q と $\bar{Q}$ をテスト生成制約回路で生成することによって，無効状態を含まないテストパターンが組合せ回路に印加される．$\bar{Q}$ は Q の反転値であるため，状態レジスタから組合せ回路までを接続している信号線を削除し，テスト生成制約回路を通った Q の信号線から NOT ゲートを使用することで，テスト生成制約回路の出力先に $\bar{Q}$ 移動させて組合せ回路に接続する．

また，テスト生成制約回路はテスト生成に制約を与えるものであり，決してスキャン回路に合成するのではない．遅延故障を検出するテスト生成を行う際も，付加したテスト生成制約回路に該当する故障は取り除く．テスト生成法に制約を与える手法なので，テスト生成制約回路を通して出力されたテストパターンは，状態レジスタ部分で出力された論理値を有効状態に変換する．

4. 実験

本節では，テスト生成制約回路を用いた遅延故障テストパターン生成という提案手法に基づいた実験を行う．通常動作に近い振る舞いをする高品質な遅延故障テストでパターンオーバーテストや電力オーバー/アンダーテストの歩留り劣化問題を緩和することができるかどうか，提案手法の有効性を示す．

4.1 実験環境

実験環境を表 3 に示す．また，実験対象としてベンチマーク回路，実験器具として米 synopsys 社の CAD ツールを用いた．

表 3 実験環境

CPU	3.0GHz AMD Opteron256(1MB キャッシュ)×4
メモリ	16GB
OS	Solaris 10
言語	VHDL(RTL)，Verilog(ゲートレベル)
論理合成ツール	Design Compiler(Synopsys)
ATPG ツール	TetraMAX(Synopsys)
論理シミュレーションツール	VCS-MX(Synopsys)
電力解析ツール	Prime Time PX (Synopsys)
使用ライブラリ	nangate45nm

ベンチマーク回路はすべてコントローラー部分に無効状態を含む RTL 回路を用いた．各ベンチマーク回路の特性を表 4 に示す．

表 4 RTL ベンチマーク回路特性表

回路名	状態数	無効状態数	FF 数	PI 数	PO 数	回路面積
paulin	8	2	227	32	32	8060
s_1_reset	32	12	5	8	6	330
pma_reset	32	8	5	8	8	216
s832_reset	32	7	5	18	19	372
dk16_reset	32	5	5	2	3	240

4.2 実験方法

各ベンチマーク回路から有効状態，無効状態を調べてテスト生成制約回路を作成する．次に，3 種類の回路を用いて遷移遅延故障を対象とした 2 パターンテスト生成を行う．

- 順序回路を用いた遅延故障テスト生成
- 従来法:スキャン設計回路を用いた遅延故障テスト生成
- 提案法:スキャン設計回路にテスト生成制約回路を用いた遅延故障テスト生成

3 つの回路を ATPG ツールを使用して遅延故障テスト生成を行うと，テストパターンが得られる．提案手法が通常動作時に近い振る舞いでテストを行えたかどうかを検証するために，本提案手法で得られたテストパターンと通常動作時の電力を表す順序回路のテストパターンの遷移率を計算する電力消費評価ツールを用いる．なお，比較対象としてスキャン設計のみを施した回路のテストパターンも用いる．

4.2.1 消費電力解析

テスト生成制約回路を通して得られたテストパターンを組合せ回路に印加し，その出力応答がどれほど通常動作の振る舞いに近付いているのかどうかを検証す

るために，米 Synopsys 社のコンパイル型高速シミュレーションツールと消費電力解析ツールを使用する．

HDL シミュレーションツールである VCS/Mixed-HDL シミュレータ (VCS-MX) を用いて，得られたテストパターンをテストベンチとして再利用する．VCS-MX で処理された結果は回路データのスイッチング情報として求めることができる．このままではスイッチング情報しか得られていないため，これに基づいて消費電力を計算し，提案手法が通常動作に近づいたかどうかを，消費電力量を基に検証する．これは，電力解析シミュレータツール (Primie time) を使用して，スイッチング情報に基づく消費電力値を求めることができる．図 19 に電力解析までのフローを示す．

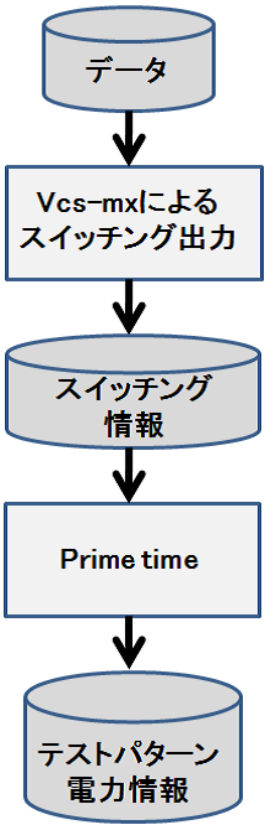

図 19　テストパターンの消費電力解析フロー

回路のスイッチング情報を作成しシミュレーションを行うことで，遷移遅延故障テストパターンの電力情報が得られる．この電力情報を解析してテスト生成制約回路によるテストパターンが通常動作に近い消費電力で動作して出力されているのかどうかを検証する．

4.3 実験結果

4 つのベンチマーク回路を用いてテスト生成と電力解析を行った．その実験結果をこの節で示す.

表 5，7，9，11 は各ベンチマーク回路を Synopsys 社の論理合成ツール (Design Compiler) と ATPG ツール (Tetra MAX) を用いてテスト生成を行った結果の比較を示す．なお，順序回路は論理合成ツールを使用していない．表 6，8，10，12 は各ベンチマーク回路のテスト生成から得られたテストパターンを Synopsys 社の電力解析ツール (Primie time) を使用して計算した消費電力の結果を比較を示す.

Scan 回路とはスキャン設計のみがなされた回路に対して生成したテストパターンを用いた場合，TGCC 回路 (Test Generation Constraint Circuit) は本論文の提案手法であるテスト生成制約回路を付加した回路に対して生成したテストパターンを用いた場合のことである.

また，図 20〜27 は各回路のテストパターンの Launch，capture 部分を比較対象とした各電力消費値に対するパターン数の分布と，順序回路の通常動作時の電力消費値に対するパターン数の分布を視覚的に比較したグラフである．ただし，順序回路のパターン数は Scan 回路と TGCC 回路と比べて非常に多いため，この 2 回路のパターン数の平均数と同じになるように比率を合わせた順序回路のパターン数に変換している.

表 5 s1_reset 回路の各テスト生成結果

s1_reset 回路	順序回路	Scan 回路	TGCC 回路
回路面積	330	336	340
総故障数	2318	2332	2328
検出された故障数	2064	2278	2277
UD 数	0	0	3
AU 数	29	54	48
ND 数	225	0	0
故障検出率 (%)	89.04	97.68	97.38
ATPG 効率 (%)	90.29	100	100
テストパターン数	2432	113	117
無効状態を含むテストパターン数	-	65	0
計算時間 (s)	5463	0.35	0.16

表 6 s1_reset 回路の各テスト生成の消費電力解析結果

消費電力 (mw)	通常動作時 (mw)	Scan 回路 (mw)		TGCC 回路 (mw)	
		Launch	Capture	Launch	Capture
平均値	0.0079	0.0070	0.0070	0.0069	0.0070
最大値	0.0107	0.0093	0.0084	0.0090	0.0102
最小値	0.0023	0.0038	0.0050	0.0027	0.0040

表 7 pma_reset 回路の各テスト生成結果

pma_reset 回路	順序回路	Scan 回路	TGCC 回路
回路面積	216	222	226
総故障数	1474	1488	1482
検出された故障数	1440	1411	1177
UD 数	0	0	270
AU 数	27	77	35
ND 数	7	0	0
故障検出率 (%)	97.69	94.83	79.42
ATPG 効率 (%)	99.53	100	100
テストパターン数	2485	190	124
無効状態を含むテストパターン数	-	47	0
計算時間 (s)	689	0.16	0.11

表 8 pma_reset 回路の各テスト生成の消費電力解析結果

消費電力 (mw)	通常動作時 (mw)	Scan 回路 (mw)		TGCC 回路 (mw)	
		Launch	Capture	Launch	Capture
平均値	0.0045	0.0059	0.0055	0.0057	0.0054
最大値	0.0073	0.0090	0.0081	0.0082	0.0080
最小値	0.0014	0.0032	0.0035	0.0032	0.0030

表 9 s832_reset 回路の各テスト生成結果

s832_reset 回路	順序回路	Scan 回路	TGCC 回路
回路面積	372	378	386
総故障数	2650	2666	2662
検出された故障数	2469	2595	2591
UD 数	0	0	6
AU 数	24	71	65
ND 数	157	0	0
故障検出率 (%)	93.17	97.34	97.33
ATPG 効率 (%)	94.48	100	100
テストパターン数	5387	408	430
無効状態を含むテストパターン数	-	18	0
計算時間 (s)	5644	0.44	0.44

表 10 s832_reset 回路の各テスト生成の消費電力解析結果

消費電力 (mw)	通常動作時 (mw)	Scan 回路 (mw)		TGCC 回路 (mw)	
		Launch	Capture	Launch	Capture
平均値	0.0083	0.0091	0.0092	0.0090	0.0091
最大値	0.0114	0.0127	0.0081	0.0132	0.0121
最小値	0.0025	0.0050	0.0038	0.0046	0.0037

表 11 dk16_reset 回路の各テスト生成結果

dk16_reset 回路	順序回路	Scan 回路	TGCC 回路
回路面積	234	240	248
総故障数	1586	1600	1592
検出された故障数	1550	1547	1539
UD 数	0	0	2
AU 数	23	53	51
ND 数	13	0	0
故障検出率 (%)	97.73	96.69	96.67
ATPG 効率 (%)	98.18	100	100
テストパターン数	2323	190	188
無効状態を含むテストパターン数	-	0	0
計算時間 (s)	526	0.11	0.15

表 12 dk16_reset 回路の各テスト生成の消費電力解析結果

消費電力 (mw)	通常動作時 (mw)	Scan 回路 (mw)		TGCC 回路 (mw)	
		Launch	Capture	Launch	Capture
平均値	0.0057	0.0075	0.0078	0.0074	0.0078
最大値	0.0086	0.0112	0.0102	0.0106	0.0102
最小値	0.0013	0.0037	0.0038	0.0031	0.0030

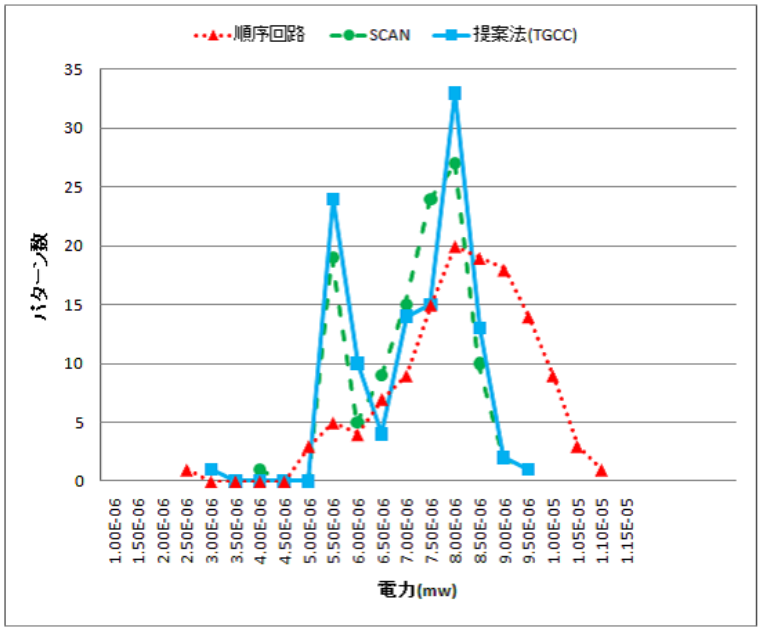

図 20　s1_reset 回路の各テスト生成 Launch 部分の電力解析結果

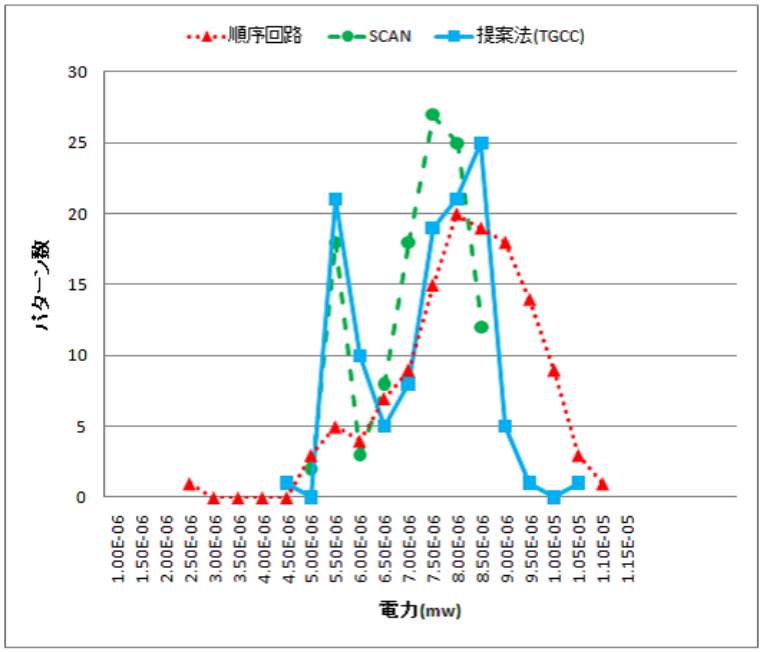

図 21　s1_reset 回路の各テスト生成 capture 部分の電力解析結果

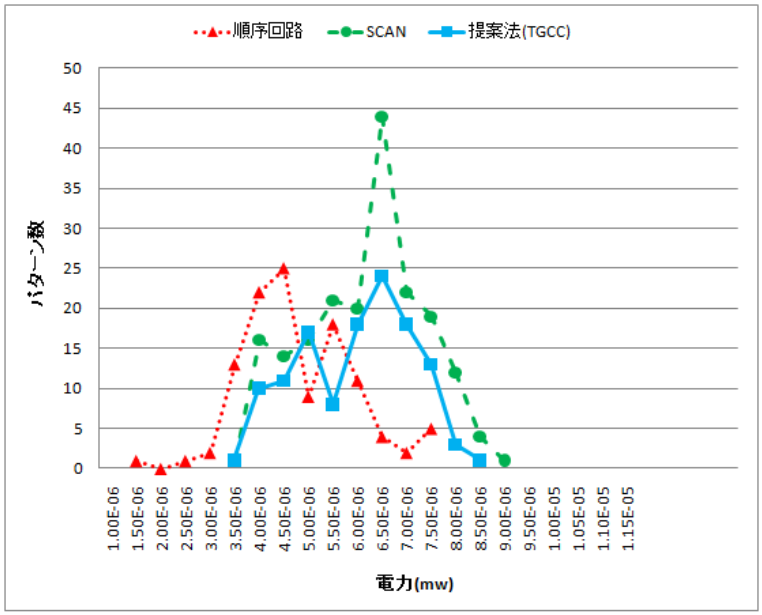

図 22　pma_reset 回路の各テスト生成 Launch 部分の電力解析結果

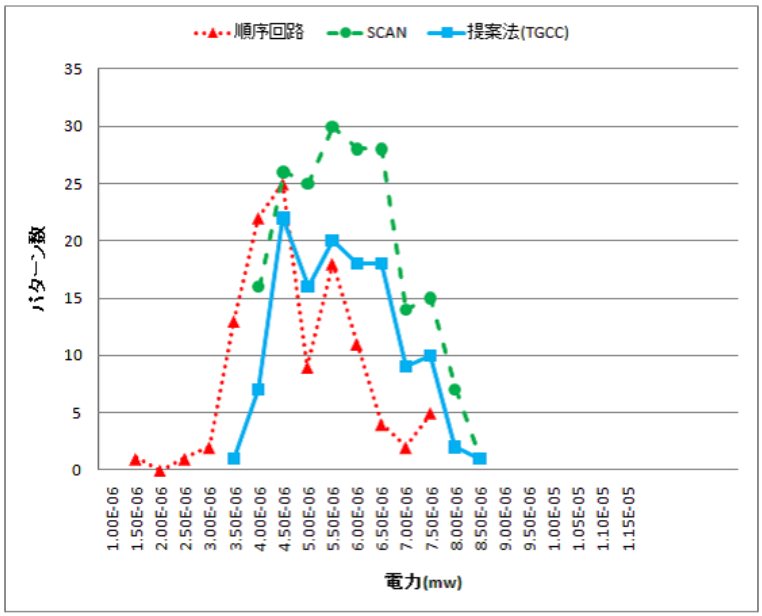

図 23　pma_reset 回路の各テスト生成 capture 部分の電力解析結果

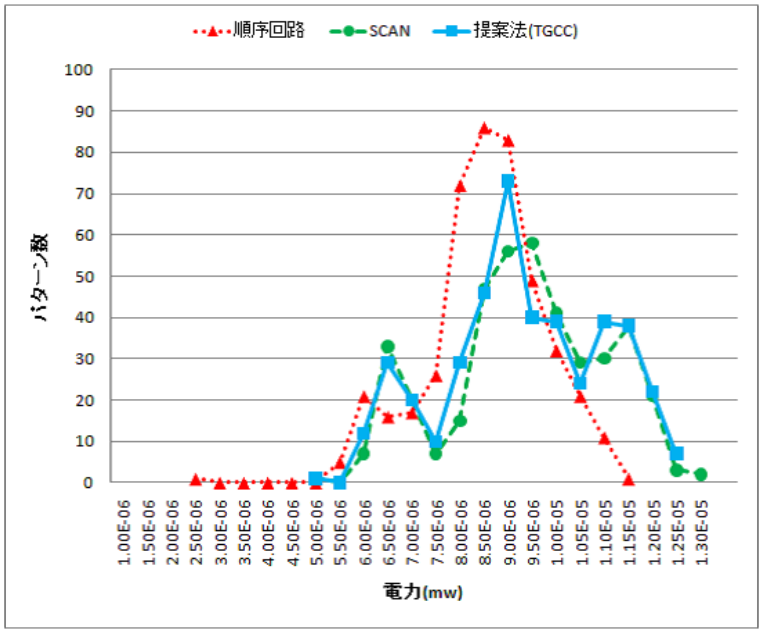

図 24　s832_reset 回路の各テスト生成 Launch 部分の電力解析結果

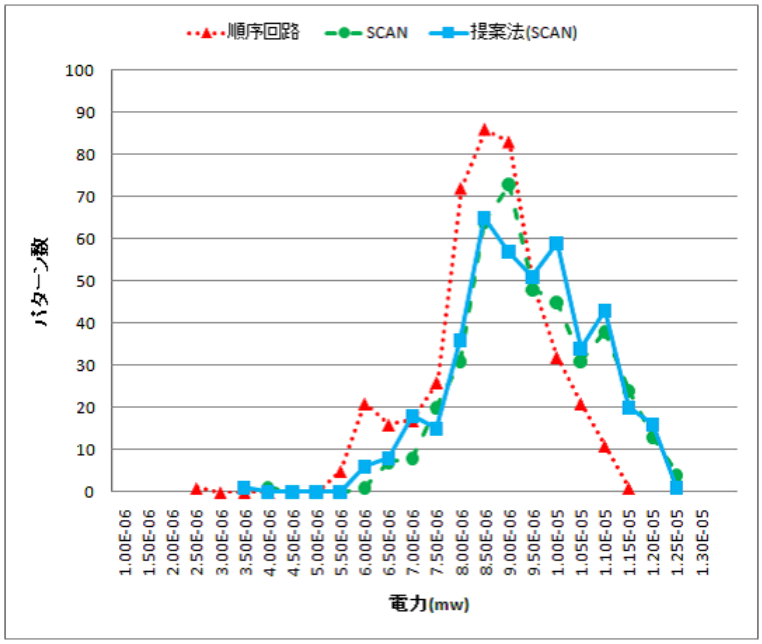

図 25　s832_reset 回路の各テスト生成 capture 部分の電力解析結果

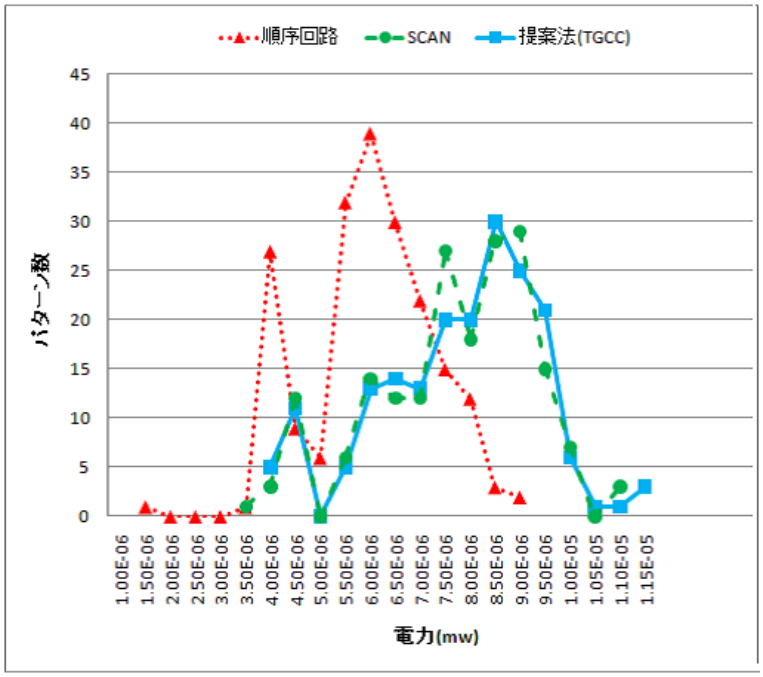

図 26　dk16_reset 回路の各テスト生成 Launch 部分の電力解析結果

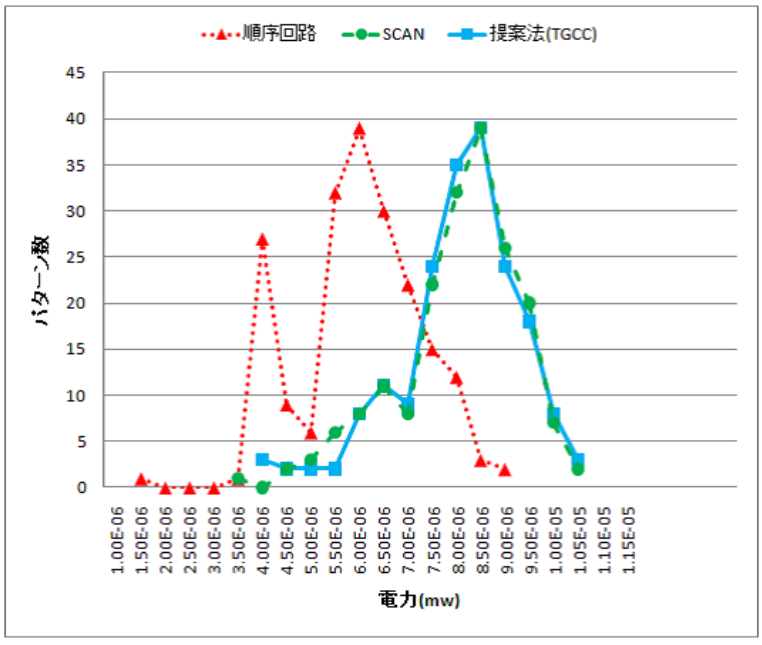

図 27　dk16_reset 回路の各テスト生成 capture 部分の電力解析結果

4.4 考察

実験結果より，パターンオーバーテストの緩和に関して，全ての回路において冗長故障数が増加しており，従来の SCAN 設計法よりもパターンオーバーテストを回避する事ができた．また，pma_reset 回路では 270 個の冗長故障を判定することができている．提案法はパターンオーバーテストを緩和することができたと言える．

また，電力オーバー/アンダーテストの緩和に関しては，s_1_reset 回路の提案手法によるテスト生成結果が，最も通常動作に対して近い振る舞いでテストパターンを生成できていることがわかった．従来法と提案法の Launch，Capture 部分の消費電力最大値/最小値を比較すると，s1_reset 回路において電力オーバー/アンダーテストの緩和，pma_reset 回路において電力オーバーテストの緩和，s832_reset 回路において電力オーバー/アンダーテストの緩和，dk16_reset 回路において電力オーバーテストの緩和ができたと言える．無効状態数が最も多い s1_reset 回路では，参考にした順序回路の消費電力の最大値をオーバーしておらず，最小電力値を下回らずに，通常動作時と近い必要十分な電力を消費してテストが行えたと言える．

提案法は幅広い範囲で電力を消費しているため電力オーバー/アンダーテストを緩和することができたと言える．

5. 結論

本研究では，まず，VLSI 設計の手順を説明した．回路を構成する要素に物理的欠陥があれば, 回路が正しい動作をしなくなりこれを故障と呼ぶ．VLSI の大規模化，高集積化に伴い，設計した回路の故障の有無を調べるテストがますます重要になってきている．デジタルシステムの構成要素は論理回路であり, 論理回路の論理機能が故障により別な論理機能に変化してしまう故障を論理故障という. これに対して, 規定時間内に故障が伝搬しない故障が遅延故障であり，VLSI の微細化や SoC の高速化に伴い，高品質な遅延故障テストの重要性が高まっている．

スキャン設計における問題としてパターンオーバーテストと電力オーバー/アンダーテストを挙げた．スキャンフリップフロップに任意の値が印加されることで，通常では設定できないパターンを第 1 パターンに用いられて無効状態を含んだテストが行ってしまう，通常動作では活性化されない故障を検出するパターンオーバーテスト問題である．遅延故障を検出するテスト方法として 2 パターンテスト生成法を挙げたが，このテストパターンを印加した際には回路全体に大規模なスイッチングが発生する．遷移遅延故障では，テスト時に想定以上の電力が消費されてしまい，テスト時にのみ一時的な遅延の増加が起き，意図しないタイミング違反 [3] が発生する．消費電力が大きくなると IR ドロップによって生じる通常動作では起こらない遅延増加により，遅延故障を誤検出する電力オーバーテスト問題，消費電力が小さくなるとテスト時に必要とする通常の厳しい電力状況による遅延増加を再現できず，遅延故障を見逃してしまう電力アンダーテスト問題である．

通常動作を考慮したテスト生成法はいくつか提案されており，制約回路を N フレーム展開する N フレーム展開モデル手法と，無効状態を考慮する X-filling 手法である．しかし，N フレーム展開モデルをテスト生成制約回路として使用する手法には，テスト対象とする回路面積が大きいと制約回路も大きくなるため，テスト生成に膨大な時間がかかってしう欠点がある．また，X-filling 手法ではゲートレベルで無効状態を解析するため，時間がかかってしまう．

そこで，本研究ではまず，順序回路を RTL レベルで解析してコントローラー部とデータパス部に切り分け，コントローラー部に着目した．一般的に，コント

ローラー部の状態から初期状態から到達可能な状態である有効状態，到達不可能な状態である無効状態を発見することができる．これはRTL 情報を用いてコントローラから高速に無効状態を抽出することができる．そして，無効状態から小さな組合せ回路から構成されたテスト生成制約回路を設計し，テスト生成の高速化を実現させた．また，通常動作時には有効状態のみが使用される．テストパターン生成に有効状態を使用して通常動作に近いテスト生成を行うことの有用性を検証するため，提案手法では，無効状態が生成されたテストパターンに表れないように，テスト生成制約回路を用いてテスト生成アルゴリズムに制約を与えた．これにより，第1パターンに無効状態が現れないようにすることができる．こうして，通常動作に近いテストパターン生成が実現できる手法としてLaunch-off-capture に基づくテスト生成法を提案した．

本論文における実験では，高品質な遅延故障テストを実現させるために，パターンオーバーテスト，電力オーバー/アンダーテストの削減を目指した．遷移遅延故障検出を対象とした順序回路ATPG，スキャン設計ATPG，提案法であるテスト生成制約回路を用いたスキャン設計ATPG のテスト生成結果，そしてテストパターンのスイッチングの消費電力を計算した．提案手法の成果として，全ての回路で本来テストで検出しなくて良い故障があることを示し，パターンパターンオーバーテストを緩和することができた．また，通常動作に近い振る舞いでテストパターンを生成できることを示し，電力オーバー/アンダーテストを緩和することができた．しかしながら，企業で扱われるようなより大規模な回路に対する対応は今後の課題としたい．

本論文は，高品質な遅延故障テストを実現するために，通常動作時の振る舞いで遷移遅延故障を行うテストパターン生成法を提案した．RTL 情報を用いて高速に無効状態を抽出し，小さなテスト生成制約回路を用いてテスト生成を高速化させた．こうして，無効状態を考慮したテスト生成制約回路を用いて通常動作に近い遅延故障テストパターン生成を行った．

本提案手法は，Launch-off-capture に基づく遅延故障テスト生成法におけるパターンオーバーテスト，電力オーバー/アンダーテストを緩和するのに有効的な手法である．

謝辞

本研究，及び本論文の作成にあたり，終始懇切なる御指導，御鞭撻を賜りまし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の藤原秀雄教授に心より深謝の意を表します．また，研究において，懇篤なる激励と御指導を賜りました大竹哲史助教に厚謝の意を表します．さらに，本研究において，有益なる御助言を頂きました井上美智子准教授，米田友和助教に感謝いたします．

また，本研究を行うにあたり，多くの示唆を与えてくれたコンピュータ設計学講座の皆様と留学生の皆様に感謝いたします．

最後に，本研究を含め，大学院生活を支えていただいた両親に感謝いたします．

参考文献

[1] 藤原秀雄，ディジタルシステムの設計とテスト，工学図書株式会社，2004.

[2] Angela Krstic and Kwang-Ting Cheng, *Delay Fault Testing for VLSI Circuits*, Kluwer Academic Pub, 1998.

[3] S.Morishima, A.Takuma, S.Kajihara, X.Wen, T.Maeda, S.Hamada and Y.Sato“ On generation of transition faults test patterns in consideration of the path length in broadside testing, ” Technical Report of IEICE, DC2005-81, pp55-60, 2006.

[4] N, Nicolici and Xiaoqing Wen,“ Embedded Tutorial on Low Power Test, ” 12th IEEE European Test Symposium, 2007.

[5] Anand Raghunathan, Sujit Dey and Niraj K. Jha “ Register-transfer level estimation techniques for switching activity and power consumption, ”International Conference on Computer-Aided Design, 1996.

[6] T.Fukuzawa, K.Miyase, Y.Yamato, H.Furukawa, X.Wen and S.Kajihara, “ A transition delay test generation method for capture power reducation during at-speed scan testing , ” Technical Report of IEICE, VLD2007-71, pp.7-12, 2007

[7] E.Alpaslan, Yu Huang, Xijiang Lin, Wu-Tung Cheng, J.Dworak,“ Reducing scan shift power at RTL , ” 26th IEEE VLSI Test Symposium, pp-139-146, 2008.

[8] 宮田一希,“ 通常消費電力テストのための遅延テスト生成, ”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 修士論文，2010.

[9] Xiao Liu, Yubin Zhang, Feng Yuan and Qiang Xu,“ Layout-aware pseudo-functional testing for critical paths considering power supply noise effects, ” Design Automation and TEST in Europe, 2010.

[10] 深山正幸，HDL による VLSI 設計，共立出版，2002.

[11] W.Lafayette,“ On the generation of scan-based test sets with reachable states for testing under functional operation conditions，”Design Automation Conference，pp.928-933，2004

[12] 柴山潔，コンピュータサイエンスで学ぶ論理回路とその設計，近代科学社，1999.

[13] Mark Zwolinski，(訳：宇野俊夫，宇野みれ)，VHDL デジタル回路設計 標準講座，翔泳社，2007.

[14] Myke Predko，(訳：矢沢久雄，日向俊二)，独習デジタル回路，翔泳社，2008.