順序回路に対する短縮スキャンシフト法とテスト系列生成法に関する研究

1996年1月

樋上喜信

①

順序回路に対する短縮スキャンシフト法とテスト系列生成法に関する研究

1996年1月

樋 上 喜 信

内容梗概

論理回路のテストは，論理回路を設計･製造する上で欠くことのできない技術である．近年における，論理回路を設計･製造する技術の進歩には目覚ましいのものがあり，そのことによって複雑で大規模な回路を，小さなチップに実現することができる．しかし，そのような回路をテストすることは，非常に難しい問題となっており，特に順序回路に対するテストは，組合せ回路に比べてはるかに難しい問題として知られている．順序回路は，回路内にメモリ素子を含む論理回路であり，多くの論理回路が順序回路として分類される．従って，論理回路のテストの問題を考える上で，順序回路の問題を避けることはできない．順序回路のテストが困難であることは，実用的な計算時間で高い故障検出率を達成するテスト系列を求めることが困難であるか，または，実用的な長さのテスト系列を求めることが困難であることを意味している．つまり，あらゆる場合を尽くすことによって，100% の故障検出率を達成することは可能であるが，そのときの計算時間やテスト系列長は，回路内のメモリ素子数乗と外部入力数乗に比例する．

そこで順序回路に対しては，テストが容易になるように回路を設計する，テスト容易化設計という方法が用いられる．テスト容易化設計にも様々の方法があるが，現在最も広く用いられているのが，スキャン設計とよばれる方法である．スキャン設計を行うことによって，回路内のメモリ素子の値を，外部から制御したり観測することが容易になり，その結果，組合せ回路に対するものと同じアルゴリズムでテスト系列生成が行え，高い故障検出率の達成も容易になる．しかし，テスト系列が長くなるという欠点がある．

テスト系列が長くなると，テストにかかる時間が長くなり，1つの回路がテスタを占有する時間が長くなる．つまり，短期間に大量の回路を製造しなければならないときには，多くのテスタを必要とし，テストのコストの増大を招く．また，テスト系列が長ければ，テスタに蓄えるべき正常回路の出力応答のデータ量が多くなり，テスタに大きなメモリが必要となる．このように，テスト系列長の増大は，テスト時間やテストにかかるコストの増大を招く．

本論文では，順序回路に対して故障検出率をできるだけ低下させることなく，テスト系列を短縮する方法を提案する．まず，スキャン設計された順序回路に対して，短縮スキャンシフト法とよぶ方法を提案する．短縮スキャンシフト法では，故障検出のために必要なフリップフロップの制御と観測の必要性を調べ，その結果に従い，スキャンシフト操作をできるだけ短くする．また，スキャン設計を行う際のフリップフロップの配列を考えることによっても，テスト系列の短縮を実現する．本論文では，短縮スキャンシフトを，フルスキャン回路，パーシャルスキャン回路，リタイミング技法に応用したときの，テスト容易化設計法とテスト系列生成法を提案する．

さらに，区別系列を持つ順序回路に対するテスト系列を短縮する方法を提案する．区別系列とは，順

序回路の内部状態を外部出力で識別するための入力系列であり，これを用いることによって，仮定した全ての故障を検出するようなテスト系列の生成法が過去に提案されている．しかし，その方法では非常に長いテスト系列が必要であった．本論文では，区別系列を持つ順序回路に対して，縮退故障を対象に，短いテスト系列を生成する方法を提案する．

第１章では，本研究に関する過去の研究の成果と問題点を述べ，本研究の動機や目的，特長を述べている．

第２章ではまず，提案方法を説明する上で必要となる，論理回路のテストに関する基本概念の説明や用語の定義を行っている．次に順序回路に対するテスト系列生成法の１つである時間展開法を述べ，順序回路に対するテスト系列生成が困難であることを説明している．さらに，スキャン設計法と，スキャン設計された回路のテスト法について述べている．

第３章では，短縮スキャンシフト法の基本的概念と，第４章から第６章で述べる各提案方法に共通する技法について述べている．ここでは故障を検出するために制御や観測が必要となるフリップフロップの求め方について，できるだけテスト系列が短くなるような方法を提案している．

第４章では，フルスキャン回路に対して短縮スキャンシフト法を応用したテスト系列生成法とテスト容易化設計法を提案している．ここでは，フリップフロップの配列法やテスト系列生成法について，テスト系列が短く，しかも従来法と同程度の高い故障検出率が達成されるような方法を提案している．実験の結果を示し，提案方法の有効性を明らかにしている．

第５章では，パーシャルスキャン回路に対して短縮スキャンシフト法を応用したテスト系列生成法とテスト容易化設計法を提案している．ここでは，フリップフロップの選択と配列の方法，テスト系列の生成法について，フルスキャン回路に対する方法と異なる方法を提案している．実験の結果を示し，提案方法の有効性を明らかにしている．

第６章では，リタイミング技法を応用した短縮スキャンシフト法を提案している．ここでは，出力関数を変えずにフリップフロップの配置を変えるリタイミング技法を３つのタイプに分け，テスト系列が短くなる場合の条件について議論している．また，リタイミング技法によって得られた回路に対するテストベクトルを，変更前の回路に対するテストベクトルから求める方法を提案している．実験の結果を示し，提案方法の有効性を明らかにしている．

第７章では，区別系列を持つ順序回路に対して短いテスト系列を生成する方法を提案している．まず，過去に提案された区別系列を持つ順序回路に対するテスト系列生成法を述べ，テスト系列が長くなる理由を述べている．次に，区別系列を持つ順序回路の設計法を述べ，短いテスト系列の生成法を提案している．実験の結果を示し，提案方法の有効性を明らかにしている．

第８章では，本論文のまとめとして，提案方法とその有効性について総括し，今後の研究課題について述べている．

目次

第１章　序論

現代社会ではあらゆる分野でコンピュータの利用が進み，それにより機器の自動化や大量の情報処理が実現されている．コンピュータの利用が広がれば広がるほど，コンピュータに対する信頼性の要求は高くなり，正常に動作することが厳密に保証されなければならない．コンピュータで行われる演算処理は，論理回路による信号演算で実現されており，コンピュータに対する信頼性は，それを構成する論理回路に対する信頼性に支えられていると言っても過言ではない．従って，論理回路の正常な動作を保証をすることは大変重要であり，論理回路のテストの意義もそこにある．テストは，広い意味では，回路が仕様どおりに設計されているかどうかを調べる設計検証や，故障位置の指摘なども含めることがあるが，以下では，製造された回路に故障があるかどうかを調べることのみをテストという．

論理回路にも様々な種類があるが，論理演算を実現する回路としては，組合せ回路と順序回路に分類される．組合せ回路は，その出力関数が入力のみによって決まる回路で，順序回路は，回路内にフリップフロップ(FFと記す)などのメモリ素子を含み，出力関数が入力と内部状態によって決まる回路である．マイクロプロセッサなど，コンピュータを構成する素子の多くは順序回路に分類される．ところが，これまでのテストに関する研究において，組合せ回路に対しては，数々の優秀な方法が開発されてきたが，順序回路に対しては未解決の問題を多く残している[1]-[5]．例えば，テストの際に用いるテスト入力を求めるテスト生成の問題では，組合せ回路に対しては，Dアルゴリズム[6]以来，PODEM[7], FAN[8], CONT[9], SOCRATES[10]などのアルゴリズムが提案され，文献[11]では，数万ゲート規模の回路に対して数十秒の計算時間で仮定したすべての故障を検出するテスト入力を求めることができると報告されている．一方順序回路に対しては，組合せ回路に対するような良い結果が得られていない．順序回路と組合せ回路の違いは，順序回路には過去の入力履歴に依存した内部状態が存在することにあり，外部入出力のみによって内部状態を制御したり観測することが非常に困難な点に，順序回路のテストが困難であることの本質的な原因がある．言い換えると，任意の内部状態に遷移させるための入力系列を見つけることや，外部出力で識別可能な状態や観測可能なメモリ素子を見つけることが困難なことが原因となっている．

テストに対する要求やその評価は，テストの質とコストに分けて考えることができる．テストの質とは，どれだけ多くの故障を検出することができるか，即ち，故障の見逃しをいかに少なくするかである．現実に起こる故障をできるだけ多く含むような故障モデルを考えることは，テストの質を向上させることにつながる．また，仮定した故障のうちできるだけ多くの故障を検出するようなテスト入力を求めることも，テストの質を向上させることである．これは仮定した故障のうち，そのテスト入力で検出できる故障の割合を示す故障検出率という尺度によって評価される．テスト

のコストとしては，テスタのコストやテストに費やされる時間などが含まれる．例えば，電子ビームテスタ[12]-[15]のような，回路内部の信号線も観測できるようなテスタを必要とする場合には，通常のテストに比べてテストのコストが増大する．また，非常に大量のテスト入力を必要とするテストでは，テスト時間が長大になり，1つの回路当たりのテスタの占有時間が長くなる．また，テストの際の出力応答に対する比較データを蓄えるためのテスタのメモリ使用量も増大する．テストに対する要求としては，回路の使用者の立場からは高い質が求められ，製造者の立場からは低いコストが求められる．

回路構造が複雑になり，素子数や信号線数が増加すると，テストはますます困難になり，非常に低い故障検出率しか得られない状況が起こり得る．このような状況では，回路が与えられた後にテストのことを考えるのではなく，設計段階からテストのことを考慮に入れたテスト容易化設計という考え方が有効であり，その方法が実用化されている．例えば，回路の状態遷移や状態識別を容易にしたり[16]-[19]，テスト時の回路内部の信号値の観測を可能にする方法[32]などが提案されている．順序回路に対するテスト容易化設計法として，スキャン設計法[20]-[22]は，現在最も広く用いられている方法の一つである．スキャン設計を用いることによって，順序回路の内部状態が外部から容易に制御や観測できるようになり，結果として，順序回路のテスト生成の問題が組合せ回路のテスト生成の問題になる．つまり，順序回路であっても，高い故障検出率を達成するテスト系列を実用的な計算時間で得ることが可能になる．しかし，スキャン設計の欠点として，1)回路面積と入出力端子数の増大，2)テスト系列長の増大，3)動作速度の低下などがあり，これらの問題を解決する必要がある．

本論文では，順序回路を対象にした短縮スキャンシフト法とテスト系列生成法を提案する．研究の目的は，テストの質に関しては，高い故障検出率の達成，テストのコストに関しては，テスト系列長の短縮である．順序回路を対象にした過去の研究は，組合せ回路を対象にした研究ほど多くはなく，しかもそのほとんどは高い故障検出率の達成を目指している．一般的に，順序回路に対するテスト系列生成は難しく，また，生成できた場合でも，その系列長が長くなる場合が多い．提案方法は，従来法と同じ程度の高い故障検出率を達成し，従来法よりも短いテスト系列の生成が可能であり，その応用範囲も広いという特長を持つ．

提案法の一つは，短縮スキャンシフト法と名付けた，スキャン設計された回路のテスト系列を短縮するためのテスト方法である．短縮スキャンシフト法は，テスト系列が長くなる原因となっているスキャンシフト操作を短くすることによって，テスト系列全体を短くする方法である．スキャンシフト操作とは，スキャン設計された回路をテストする際に，FFの値を外部から直接制御や観測を行う操作のことである．従来，スキャンシフト操作においては常に全てのFFの制御と観測を行っていたが，短縮スキャンシフト法では，故障を検出するために必要となるFFを求め，その結果に応じて，不要なスキャンシフト操作を省略する．問題は，どの故障を対象にして，どのように制御や観測の必要なFFを見つけるかということである．対象故障の選択の際には，ある基準に従った検出困難な故障を選び，故障検出率を低下させることなく，テスト系列の短縮を実現する．短縮スキャンシフト法のもう一つの特徴は，レイアウト上の制約の範囲内で，短縮スキャンシフト法をより有効にするFFの配列を考える点である．スキャンシフト操作はFFの配列によっても影響を受けるため，FFの制御と観測の必要性を表す尺度を導入し，それを用いて，テスト系列が短くなるようにFFを配列する．

短縮スキャンシフト法は，スキャン設計された回路ならどのような回路にも適用でき，また，他のテスト容易化技法と併せて用いることもできる．本論文では，短縮スキャンシフト法をフルスキャン回路とパーシャルスキャン回路に適用した場合について述べる．フルスキャン回路もパーシャルスキャン回路もいずれもスキャン設計された回路の一種であるが，それぞれスキャン設計によって制御や観測可能なFFの割合が違うため，短縮スキャンシフト法を用いるときの方法や目的に違いが生じる．フルスキャン回路では，テスト系列を構成する基となるテストベクトルが与えられたとき，達成できる故障検出率の上限が分かるので，テスト系列生成においては，その故障検出率を低下させることがないような方法を採らなければならない．一方，パーシャルスキャン回路では，フルスキャン回路のような達成できる故障検出率の上限が分からないので，フルスキャン回路の場合とは違う方法によって，できるだけ高い故障検出率を達成しなければならない．また，パーシャルスキャン回路においては，スキャン設計によって変更するFFの選択についても考慮しなければならない．これらのことを考慮に入れながら，短縮スキャンシフト法によるテスト系列の短縮を行う．

最近注目されつつある論理合成及びテスト容易化設計技法としてリタイミング技法[24]-[28]があり，これを応用した短縮スキャンシフト法についても提案する．リタイミング技法を用いればFF数を減らすことができ，その結果，スキャン設計された回路のテスト系列長が短くなることは容易に理解できる．しかし，短縮スキャンシフト法を用いた場合には，たとえFF数が減らなくても，テスト系列長が短くなる場合があることを本論文で示す．テスト系列を短縮するために，どのようにリタイミング技法用いるべきか，また，どのようにテスト系列を生成すべきかについて，有効な方法を提案する．

本論文では，区別系列を持つ順序回路に対する短いテスト系列の生成法についても提案する．区別系列は順序回路の内部状態を外部から識別するための入力系列であり，これを用いた順序回路の同定問題は古くから研究されている[23]．文献[23]で提案された方法の利点は，回路の出力や状

態遷移を変えたり状態数を減らすような，どのような故障も検出可能である点と，テスト系列が状態遷移表に基づいて生成されるので，テスト系列が回路の実装方式に依存しない点である．その反面，テスト系列長が非常に長くなるという欠点があるため，提案方法では，対象故障を論理回路のテストで一般的によく用いられている縮退故障にして，テスト系列の短縮を行う．縮退故障を対象とすることと区別系列を用いることで，組合せ回路のテスト生成アルゴリズムが適用でき，そのため，テスト系列長を短く，またほとんどの回路に対して100%の故障検出率を得ることができる．与えられた回路が区別系列を持たない場合であっても，回路の記述レベルに応じて，区別系列を持つような回路に変換するテスト容易化設計法についても提案する．

論文は次のように構成されている．第2章において，順序回路モデル，故障モデル，テスト生成などに対する基本概念の説明と用語の定義を行った後，順序回路のテスト生成法の一つである時間展開法を紹介し，順序回路のテスト生成の困難さを述べる．更に，順序回路に対するテスト容易化設計の一つであるスキャン設計法とスキャン設計された回路のテスト法について説明する．第3章では，短縮スキャンシフト法の基本概念と様々な方法へ応用するときに共通して用いられる技法を述べる．第4章では，フルスキャン回路に対して短縮スキャンシフト法を用いた，テスト容易化設計法とテスト系列の短縮法を述べる．第5章では，パーシャルスキャン回路に対して短縮スキャンシフト法を用いたときの，テスト容易化設計法とテスト系列の短縮法を述べる．第6章では，リタイミング技法を応用した短縮スキャンスキャンシフト法による，テスト容易化設計法とテスト系列の短縮法を述べる．第7章では，区別系列を持つ順序回路のテスト系列の短縮法を述べる．そこでは，与えられた回路の区別系列の見つけ方，区別系列を持つ回路の設計法，区別系列を用いた回路のテスト法，テスト系列の短縮法を述べる．最後に第8章において，本論文を総括し，今後の課題について述べる．

第2章　順序回路のテスト

2．1　順序回路モデル

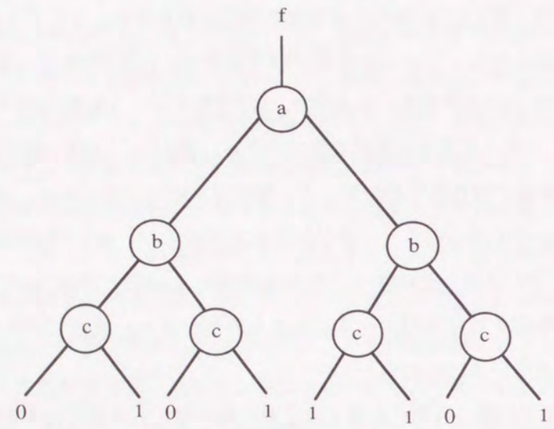
回路を表現するモデルには，回路動作を表現するモデルと，回路構造を表現するモデルがある．回路動作を表現するモデルは，扱う情報の種類によってさらに分類できる．回路動作を表すために論理値を用いたモデルには，真理値表やBDD(Binary Dicision Diagram)，状態遷移表などがある．真理値表は，表2.1に示されるような，論理値0，1によって回路の入力値と出力値の対応を表現したものである．回路動作を記述する最も単純な形式であるが，入力数nに対して2^nの大きさが必要となる．BDDは，図2.1に示されるような，回路の関数を表したグラフモデルである．BDDの各ノードは入力変数を表し，ノードから下に伸びる枝は，左側が0の入力値，右側が1の入力値を表す．また，グラフの終端が関数の出力値で，図2.1のグラフは関数$f = a\ \bar{b} + c$を表している．図2.1(a)が真理値表より得られる最も単純な形式である．図2.1(a)と図2.1(b)では，ルートから終端までどの経路上でも変数の現れる順序が同じで，順序付きBDDと呼ばれる．図2.1(c)では，変数の現れる順序が経路によって異なるので，順序付きBDDではない．図2.1(b)は図2.1(a)において，冗長や等価なノードを全て削除した結果で，既約な順序付きBDDと呼ばれる．既約な順序付きBDDは，変数の順序を固定すれば表現が一意に決まる特長があり，回路の合成や検証，テストの分野で最近広く用いられている[29]-[31]．

状態遷移表は，順序回路の動作を表現するために用いられ，入力に対する内部状態の変化と出力を表している．状態遷移表の一例を表2.2に示す．例えば表2.2では，回路の状態がq_1で，入力0を印加すると，回路の状態はq_2に遷移し，出力は0となることを表している．回路に入力が印加される直前の状態を現状態，印加した結果遷移する状態を次状態という．状態遷移表と同じ情報は，

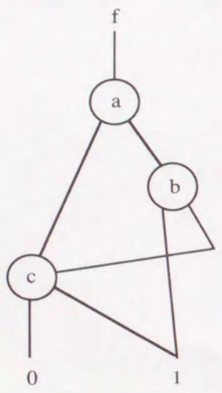
表2.1　真理値表

入力			出力
x_1	x_2	x_3	Z
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

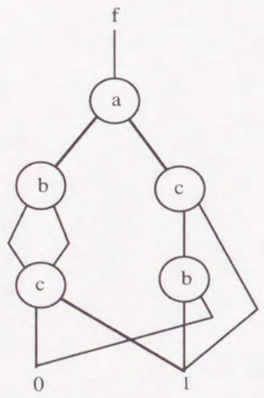
図2.2のような，状態遷移図によっても表現することができる．各ノードが状態を表し，ノード間の矢印が状態遷移を表す．矢印の元が現状態，先が次状態に相当する．状態遷移に伴う入出力値は矢印と共に表されている．順序回路が状態遷移表や状態遷移図で表されるときには，その回路は，状態の変化がクロック信号に同期して起こる，同期式順序回路であることを暗に意味している．



(a)



(b)

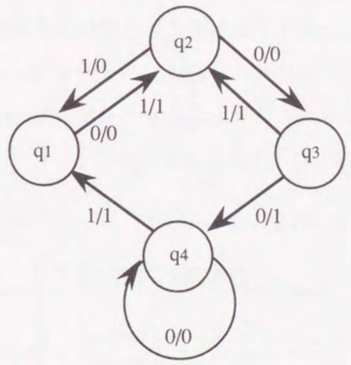


(c)

図2.1　$f = a\bar{b} + c$ の BDD

表2.2　状態遷移表

現状態	次状態 入力 0	次状態 入力 1	出力 入力 0	出力 入力 1
q_1	q_2	q_2	0	1
q_2	q_3	q_1	0	0
q_3	q_4	q_2	1	1
q_4	q_4	q_1	0	1



0/1: 入力値／出力値

図2.2　状態遷移図

回路動作を表現するために，論理値ではなく，レジスタのデータワードを用いた表現として，レジスタ転送レベルでの記述がある．例えば，レジスタAとBの和をレジスタCに代入する操作は，

$C = A + B$

と表される．条件を含む操作，もしコントロール信号Xが1なら，$C = A + B$を実行する操作は，

if X then $C = A + B$

と表される．レジスタ転送レベルは，高位の記述と呼ばれ，設計の初期の段階で，回路の機能を表すために用いられる．

回路構造を表現する記述は，ゲートレベルやトランジスタレベル，レイアウトレベルに分類される．ゲートレベルでは，AND, OR, NAND, NOR, XOR, XNOR, NOTの各論理ゲートを信号線で接続することによって回路を表現している．順序回路内のメモリー素子としては，ラッチやフリップ

フロップ(FF)を用いて記述される．FFには，RSFF，JKFF，TFF，DFFなどがある．RSFFとJKFFの機能としては，状態のセット，リセットと状態の保持が可能で，TFFは状態の反転，DFFは状態の保持が可能である．FFは，クロック信号が入力したときのみ状態変化を起こす．ゲートレベルで表された同期式順序回路モデルを図2.3に示す．FFの値の集合が，回路の内部状態であり，FFの出力に相当する方が現状態，FFの入力に相当する方が次状態となる．また，順序回路のFFを除いた部分回路は，現状態と外部入力値によって，次状態と外部出力値が一意に決まる組合せ回路の動作を行うため，その部分回路を，組合せ回路部分または組合せ部分という．FFについては，それがどのような機能のFFであっても，1種類のFFと幾つかのゲートを組合せることによって表現できる．例えば，図2.4のように，RSFFをDFFから構成することが可能である．従って，図2.3のFFをDFFに限定した場合でも，一般性を失わずに順序回路を表現することができる．

回路の実装技術に合わせて，論理ゲートをMOSトランジスタやバイポーラトランジスタなどで回路を表現したのが，トランジスタレベルの記述である．CMOSトランジスタによる2入力NANDゲートは図2.5に示される．さらにトランジスタや配線を，シリコン・アルミニウムなどで記述し

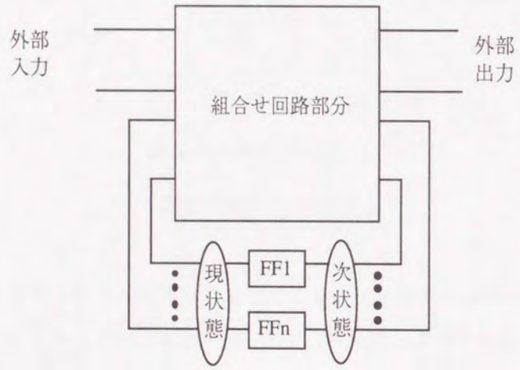


図2.3　順序回路モデル

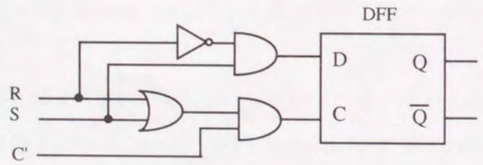


図2.4　DFFから構成したRSFF

たのが，レイアウトレベルの記述である．

回路の記述を設計の流れに合わせて表したのが，図2.6である．高位の記述では，回路の機能が，低位の記述では，回路の構造が表される．

2．2　故障モデル

回路が誤動作を起こす原因は様々考えることができる．例えば，設計段階において仕様と異なる回路を設計するエラーや，回路を構成するコンポーネントを誤って組み立てるエラーなどがある．また，製造工程において，異物が混入したために起こる短絡や断線，その他の物理的欠陥も，誤動作の原因となる．実際に起こる物理的欠陥は，回路素子の物理的特性を多岐に変化させるため，そ

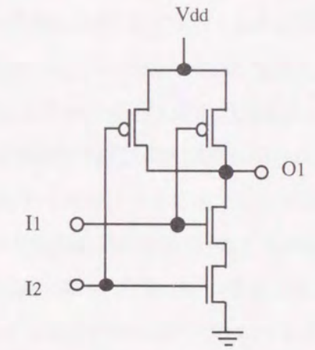


図2.5　CMOSによるNANDの記述

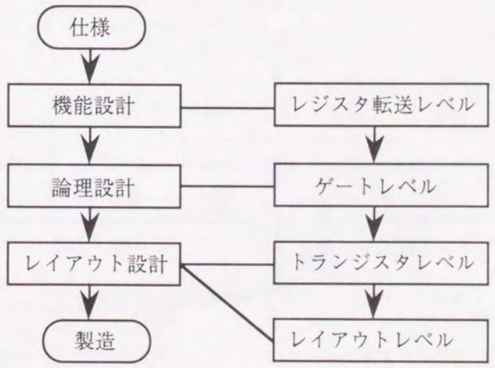


図2.6　設計の流れと回路の記述レベル

れらを忠実に記述することは困難であり，また，テストの立場からそれはあまり意味を持たない．実際に起こる物理的欠陥が，回路や素子の着目すべき特性をどのように変化させたかが重要であり，それを表したものが故障モデルである．故障モデルは，多くの種類に分類可能で，例えば，論理故障とパラメータ故障，固定故障と間欠故障などがある．論理故障は，それが存在しても論理機能は保たれる故障，パラメータ故障は，電圧・電流特性などのパラメータが変化し論理機能が失われる故障である．固定故障は，その存在が永続する故障，間欠故障は，ある条件が揃った場合の一定期間のみ顕在化し，その存在が永続しない故障である．一般によく用いられる故障モデルを回路の記述レベルに合わせて分類することもでき，それを表2.3に示す．回路の記述が詳細になれば，それに対応する故障モデルは，現実の物理的欠陥をより忠実に表現できるが，反面，素子数が増大するためテストは困難になる．逆に，高位の記述では，その故障モデルが，現実に起こりうる故障にどれほど対応しているかが問題となる．テストに関する過去の研究においては，ゲートレベルで表された縮退故障を対象にするものが多く研究されてきた．しかし近年では，CMOS回路で問題となるスタックオープン故障や，高速動作環境で問題となる遅延故障などの研究も盛んである．また，高位の回路記述から自動的に回路を設計する技術が開発されるにつれ，高位の故障モデルによるテストも，研究されつつある．

本論文では，信号線の論理値が0または1に固定する縮退故障を対象とし，しかも，回路内の一箇所にしか故障の存在しない単一縮退故障モデルを扱う．縮退故障が古くから広く用いられているのは，1)多くの異なる物理的欠陥を表す[33]，2)回路実装技術に依らず信号線の値で表現される[2]，3)経験的に縮退故障を対象とすることが他の故障の検出につながる[2]，4)多重故障の組合せは膨大で，それらを全て扱うことは現実的には不可能である，5)ほとんどの多重故障は単一故障を対象にしたテストで検出される[2]，といった理由による．

表2.3　回路の記述と故障モデル

回路記述	故障モデル
レジスタ転送レベル	機能故障
ゲートレベル	縮退故障，遅延故障，ブリッジ故障
トランジスタレベル	スタックオープン故障
レイアウトレベル	断線，抵抗・容量の変化

2．3　テスト生成

テストの際に印加する入力を，テストパターンまたはテストベクトルという．順序回路のテストでは，テストベクトルを印加する順序が意味を持ち，そのような順序付けされた複数のテストベクトルをテスト系列という．テストベクトルやテスト系列の質は，故障検出率という，仮定した故障のうちどれだけを検出することができるかを表す尺度により評価される．また，ある故障検出率を達成するために必要なテストベクトル数やテスト系列長も重要な評価尺度である．ある故障モデルに対するテストベクトルやテスト系列を求めることを，テストベクトル生成，テスト系列生成または単にテスト生成という．テスト生成では，故障のない回路と故障を仮定した回路で，出力値が異なるような入力を求める．回路と対象故障リストと達成すべき故障検出率が与えられたときの，一般的なテスト生成の処理の流れは，図2.7に示される．ここで，故障シミュレーションとは，何らかの方法で得られた入力ベクトルまたは入力系列で検出することのできる故障を見つける操作で，検出できると分かった故障は，故障リストから削除される．故障シミュレーション法については，次節で述べる．

テスト入力を求める最も簡便な方法は，乱数を用いてテストベクトルを発生させ，そのテストベクトルで検出できる故障を求める方法である．この方法では，テスト入力を求めるための計算時間は非常に短くてすむ．しかし，高い故障検出率を得たい場合や，テストベクトル数を少なくしたい場合などには有効でない．高い故障検出率や少ないテストベクトルを得るために，アルゴリズムにより，対象故障を確実に検出するテスト入力を求める方法がある．例えば，組合せ回路を対象にし

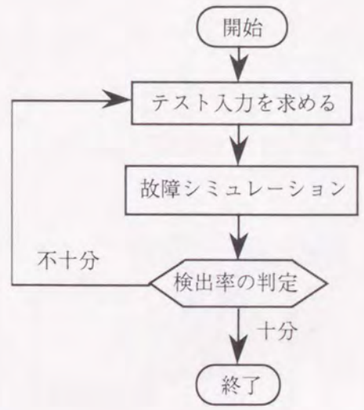


図2.7　テスト生成の流れ

た，Dアルゴリズム[6]，PODEM[7]，FAN[8]，CONT[9]，SOCRATES[10]などが提案されている．これらの方法は，対象故障を検出するために，信号線に値を割り当てていき，途中で矛盾が生じたときにバックトラックを起こすなどの，探索問題の解法を応用している．そこでは，バックトラックの発生をいかに少なくするかが，計算時間短縮の鍵であり，そのために様々な工夫をしている．計算時間を短縮することは，計算途中で処理を諦める故障の数が少なくなることを意味し，結果として，ほとんどの故障に対して，テストベクトルを生成できるか，あるいは，それが検出不能故障であると判断できるようになる．ただし，順序回路に対しては，組合せ回路に対する程，有効なアルゴリズムが開発されておらず，高い故障検出率を得られない場合がある．

２．４　故障シミュレーション

故障シミュレーションは，テスト生成を効率的に行う上で欠くことのできない処理である．順序回路に対する故障シミュレーションを高速に実行することは，テスト生成の場合と同様，組合せ回路に対する場合より困難な問題である．表2.4は，m個の故障$f_1, f_2, ..., f_m$を対象に，v個のベクトル

表2.4　故障シミュレーションの計算

	t_1 ・・・	t_j ・・・	t_v
Good	G_1	G_j	G_v
f_1	$F_{1,1}$	$F_{1,j}$	$F_{1,v}$
f_2	$F_{2,1}$	$F_{2,j}$	$F_{2,v}$
f_i	$F_{i,1}$ ・・・	$F_{i,j}$ ・・・	$F_{i,v}$
f_m	$F_{m,1}$	$F_{m,j}$	$F_{m,v}$

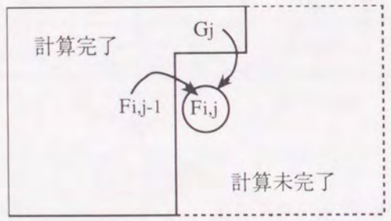


図2.8　演繹法と同時法による計算

$t_1, t_2, ...t_v$からなるテスト系列に対する故障シミュレーションで計算すべきものを概念的に表している．計算すべきは，入力ベクトルt_j ($j = 1, 2, ..., v$) に対して，正常な出力値G_j ($j = 1, 2, ..., v$) と，故障f_iの存在下での出力値$F_{i,j}$である．

演繹法[34]と同時法[35],[36]はいずれも，$F_{i,j}$を計算する際に，正常な出力値G_jと，前の入力ベクトルに対する同じ故障存在下の出力$F_{i,j-1}$を利用して計算する(図2.8参照)．この方法の利点は，正常値と異なる故障だけを計算するので，計算量が少なくてすむことである．しかし，計算のために大きな記憶容量を必要とする．演繹法と同時法の違いは，内部信号線において正常値と値の異なる故障を計算する際に，演繹法では，故障の集合演算を行い，同時法では，正常値と故障値の比較を行う点である．

差分法[37]は，図2.9に示されるように，同じ入力ベクトルで異なる故障に対する結果$F_{i-1,j}$から，$F_{i,j}$を計算する．この方法は，同時法のように内部信号線で正常値と比較しないので，内部信号線における故障情報や正常値の情報を記憶する必要がなく，外部出力線とFFの入力線だけの記憶ですむ．従って，記憶容量が少なくなる．反面，外部出力において，検出できた故障を見つけるための計算が必要となる．

並列法 [38] , [39] は，図2.10のように，計算機の１ワードの長さに合わせて，複数の故障を同時に計

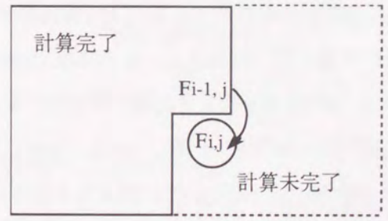


図2.9　差分法による計算

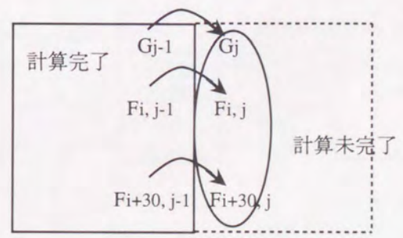


図2.10　並列法による計算

算する方法である．例えば32ビットの計算機なら，前の入力ベクトルに対する結果を利用して，正常値G_jと31個の故障に対する出力値$F_{i,j}$から$F_{i+30,j}$を同時に計算することができる．ただし，故障値を計算する際には常に，正常値を計算しなければならない．

２．５　時間展開法

同期式順序回路のテスト系列生成法の一つである時間展開法を説明する．時間展開法 [40]-[45] は，図2.11のように順序回路を時間展開し，フィードバックループをなくすことによって，組合せ回路に対するテスト生成と同様の方法を用いるものである．同期式順序回路では，全てのフィードバックループ上に少なくとも1つのFFを持つため，FFの部分でそのループを切断し，連続する2つのクロック信号の間の回路の動作を1つのセルで表すことができる．このようなセルを複数つなげることによって，順序回路の動作を組合せ回路の動作として表したのが，時間展開モデルである．各セルは時刻 i で識別され，時刻 i の入力と状態(現状態)が，時刻 i での出力と，次状態，即ち，時刻 i+1 の状態を決定する．テスト生成を行う際には，1)故障の顕在化，2)故障の影響の伝搬，3)状態の初期化を，時刻を進めながら入力を生成する前方時間操作と，時間を逆に進める後方時間操作によって求める．ここで，状態の初期化とは，回路を未知の状態から既知の状態に遷移させることで，初期化を行う入力系列を初期化系列とよぶ．一般にテストを開始する際の回路の状態は未知であることから，このような初期化が必要となる．

具体例を用いて説明する．図2.12の回路のゲートG_4の1縮退故障に対するテスト系列を求める．図2.13は図2.12の回路を時間展開したものである．まず時刻0で故障を顕在化する入力ベクトルと状態を求める．ゲートG_4の出力が0となったとき故障が顕在化するので，$I_2 = 0, FF_1 = 0$ が決まる．ゲートG_4の信号値0/1は，正常値0と故障値1を表している．次に，故障の影響がゲートG_6から次状態のFF_1に伝搬するように，G_6のもう一方の入力を1にするような，$FF_2 = 0$ を得る．結果として，まず時刻0で故障を顕在化する入力ベクトル$I_2 = 0$と状態 (FF_1, FF_2) = (0, 0) が得られる．この後

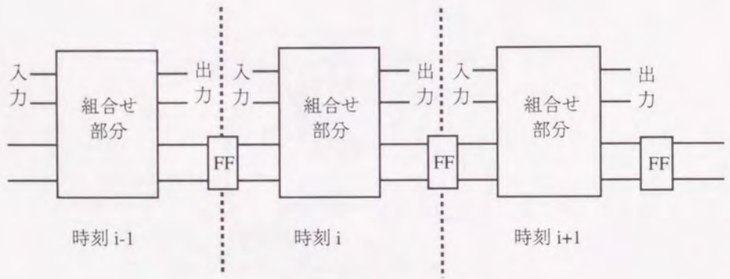


図2.11　順序回路の時間展開モデル

は，故障の影響が外部出力に伝搬するように，時刻を進めて計算する前方時間操作と，状態 (FF_1, FF_2) = (0, 0)に初期化するための後方時間操作を行う．

前方時間操作では，時刻1において，FF_1に伝搬した故障影響を外部出力O_1に伝搬させるような入力ベクトルを求める．FF_1からO_1への経路上のゲートG_2によって故障の伝搬が阻止されないようにするためには，$I_1 = 1$が必要で，これが時刻1での入力ベクトルとなる．後方時間操作では時刻-1において，外部入力の値だけで，時刻0の状態 (FF_1, FF_2) = (0, 0)を実現することを目指す．時刻0でFF_1を0にするには，ゲートG_3とG_4の出力を1にすることが必要となる．G_3を1にするには，I_2を1にするか，FF_2を0にするかのいずれかであるが，初期化の目的上，$I_2 = 1$ を選択する．この結果，

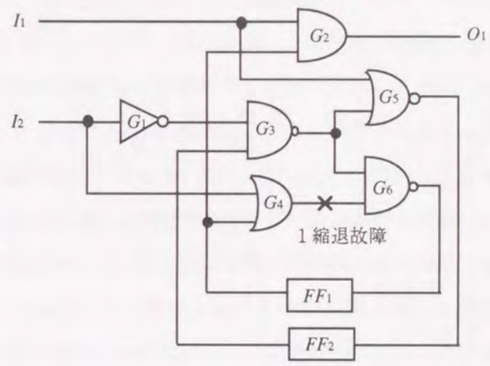


図2.12　順序回路例

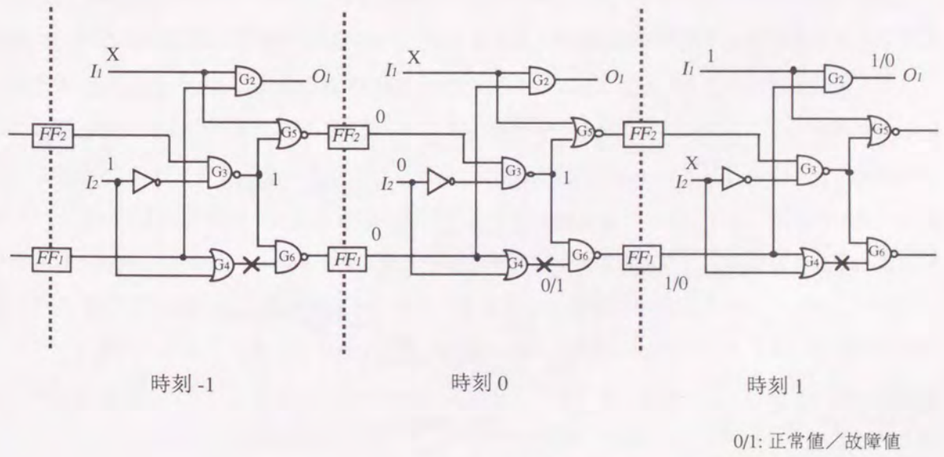


図2.13　回路の時間展開とテスト生成

G_4の出力は自動的に1になり，時刻0のFF_1は0となる．また，$I_2 = 1$のとき，時刻0のFF_2は自動的に0になる．従って，時刻-1で$I_2 = 1$を印加すれば，現状態に拘わらず，次状態として(FF_1, FF_2) = (0, 0)が実現される．つまり，$I_2 = 1$が初期化系列となる．

図2.12の回路に対する時間展開法によるテスト系列生成の結果をまとめると，最初に$I_2 = 1$，次に$I_2 = 0$，その後$I_1 = 1$を印加するような入力系列が，ゲートG_4の1縮退故障に対するテスト系列となる．

次に時間展開法における問題点を指摘する．1つ目は，テスト系列を求めるために必要となる時間展開の長さ(セルの数)の上限を予想することが困難な点である．前方時間操作と後方時間操作の両方において，最大どの時刻まで進めるか，または戻るかを予め予想することが困難である．理論的にはFF数nに対して4^nという長さが与えられるが，これは実用的ではない[2]．時間展開の長さの上限が分からなければ，計算途中で起こるバックトラック処理に対する必要な情報の大きさが分からず，結果として，アルゴリズム上すべての場合を尽くすことが保証できず，与えられた故障に対するテスト系列が存在するかどうかを判別できないことになる．2つ目の問題点は，元の順序回路で単一故障を仮定したときでも，時間展開法においては多重故障として扱わなければならない点である．つまり，時間展開したモデルの各セルの同じ位置に，故障の存在を考えなければならない．図2.13の例では，各時刻毎に故障を仮定してもうまくテスト系列が生成できたが，場合によっては，故障の影響が伝搬する際にもう一度故障が顕在化し，それによって故障の影響が打ち消し合うことが起こる．3つ目は，故障の存在下で状態の初期化を行うことが困難な点である[50], [86]．例えば図2.14の回路において，初期化系列を求めることを考える．回路に故障がない場合，(I_1, I_2) = (0, 0) が初期化系列となり，FFは0に初期化される．しかし，ゲートG_1の出力線の1縮退故障が存在したときには，初期のFFの値をXとすると，ゲートG_4の出力線の値はどのような入力を加えても0，1にはならず，即ちFFを0または1に初期化することができない[50]．

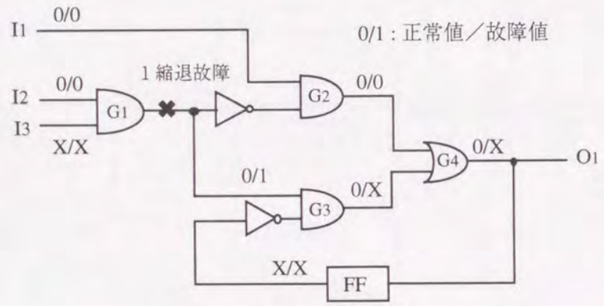


図 2.14　初期化のできない回路

時間展開法は，組合せ回路に対するテスト生成法を応用した方法であるが，解決すべき問題も残っており，数千ゲート規模の回路でさえ，回路内のすべての単一縮退故障に対するテスト系列を生成できないのが現状である．

２．６　スキャン回路のテスト

順序回路に対するテスト容易化設計法の1つとしてスキャン設計[20]-[22]がある．スキャン設計とは，回路内のFFを外部入力から外部出力に至る1本の信号線で直列につなぐことによって，FF内部の値を外部から制御や観測できるように設計する方法である．図2.15はスキャン設計で用いられるFF(スキャンFF)の一例であり，それを直列に接続した構造をスキャンチェーンといい，図2.16に示す．スキャン設計された回路(スキャン回路)のうち，回路内の全てのFFをスキャンFFにしてスキャンチェーンを構成した回路をフルスキャン回路，一部のFFをスキャンFFにしたものをパーシャルスキャン回路という．スキャンFFの入力Dは，組合せ回路部分からのデータ入力線，SDは，スキャンチェーン上の1つ前に配列されたFFまたはスキャン入力からのスキャンデータ入力線，MCは動作モードの制御線である．MCが1のとき，FFにはDからの値が取り込まれ，回路は通常の順序回路の動作を行う．MCが0のとき，FFにはSDからの値が取り込まれ，クロック信号CLが印加される度にスキャンチェーン上のFFをスキャン入力からスキャン出力へ値がシフトする(図2.17参照)．スキャンデータを取り込んで順次シフトさせる操作をスキャンシフト操作という．

スキャン回路では，状態の制御と観測が可能であるため，スキャンシフト操作によって任意の状態に回路を初期化することができ，FFに伝搬した故障の影響もスキャンシフト操作により確実に外部出力に伝搬できる．即ち，テストベクトルとしては，故障を顕在化し，その影響を外部出力またはFFに伝搬しさえすればよく，順序回路のテスト生成の問題が組合せ回路の問題と等価となり，順序回路固有の初期化や多重故障の問題を考える必要がない．故障を顕在化し，その影響を外部出力またはFFに伝搬するような外部入力値と回路の内部状態を表すベクトルを，スキャン回路のテストベクトルまたはスキャン回路の組合せ部分に対するテストベクトルといい，回路内の全ての故障を対象に生成されたテストベクトルの集合をテスト集合という．また，故障が顕在化し，その影響が外部出力またはFFに伝搬したとき，その故障は組合せ回路的に検出されたという．

スキャン回路のテストでは，スキャンシフト操作を用いて回路の状態をテストベクトルで要求された状態に設定した後，テストベクトルの外部入力値を印加する．外部出力に伝搬した故障の影響は，テストベクトル印加直後に観測され，FFに伝搬した故障の影響は，スキャンシフト操作を行うことによって観測される．スキャンシフト操作によりFFを観測するとき，同時に，次のテスト

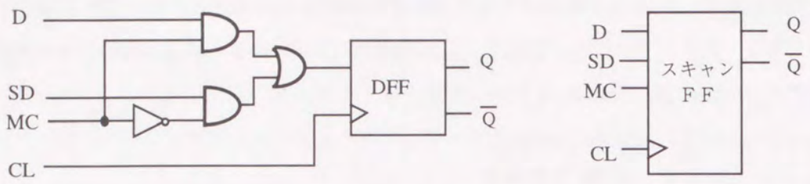


図2.15　スキャンＦＦ

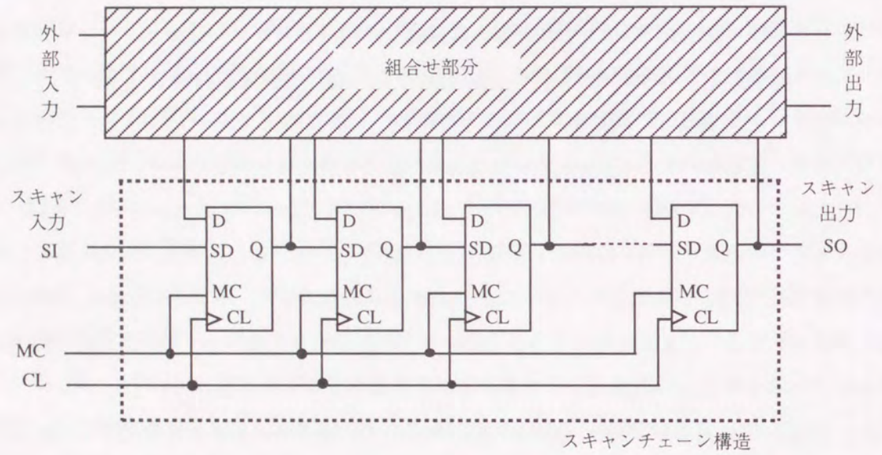


図2.16　スキャンチェーンとスキャン回路

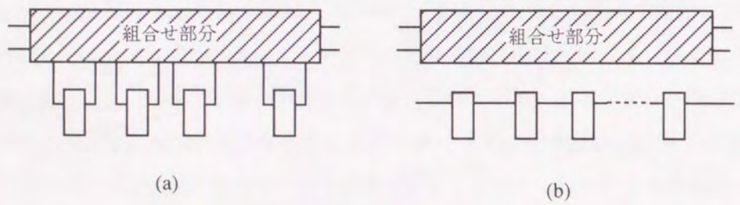


図2.17 (a) 通常動作　(b) スキャンシフト動作

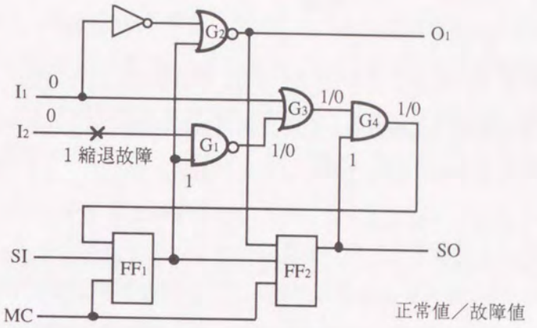


図2.18　スキャン回路

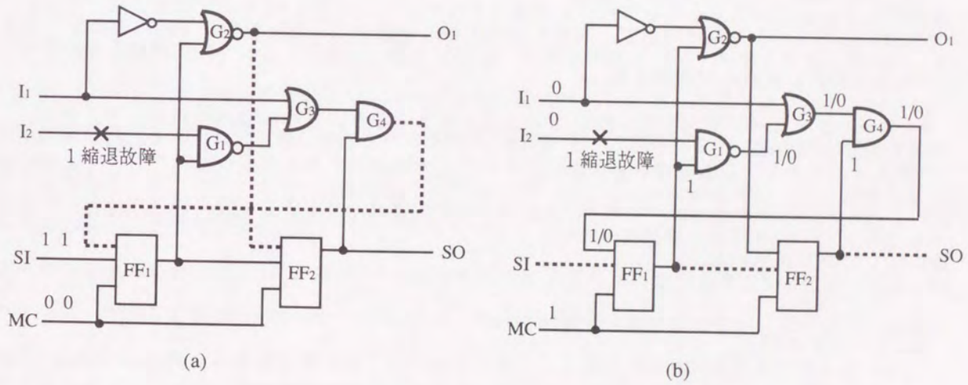


図2.19　(a)スキャンシフト操作　(b)テストベクトルの印加

ベクトルで要求される状態を設定する．従って，スキャン回路のテスト系列長は，スキャンFF数をn，その回路をテストするために必要なテストベクトル数をvとしたとき次のように表される．

$$L = v(n+1) + n \qquad (2.1).$$

図2.18の回路の信号線I_2の1縮退故障に対するテスト生成と，スキャンシフト操作を用いたテスト法について図2.19を用いて説明する．テスト生成ではまず故障を顕在化するために，$I_2 = 0$ が決まる．次に故障の影響が伝搬するように，伝搬経路上のゲートの他方の入力をゲートの非制御値に決めていくと，FF_1=1, I_1=0, FF_2=1が得られる．結局，与えられた故障に対するテストベクトルとして$(I_1, I_2, FF_1, FF_2) = (0, 0, 1, 1)$が得られ，故障の影響は$FF_1$に伝搬する．このテストベクトルを用いて実際にテストする際には，まず状態$(FF_1, FF_2) = (1, 1)$を設定するためのスキャンシフト

操作を行う．具体的には，信号線MCを0にし，クロック信号と共にスキャン入力から11を順に印加する．状態(FF_1, FF_2) = (1, 1) が設定できたところで，信号線MCを1にし，外部入力に(I_1, I_2) = (0, 0) を印加する．その結果故障の影響がFF_1に伝搬するので，これを観測するためのスキャンシフト操作として，信号線MCを0にし，クロック信号を2クロック分印加すると，スキャン出力SOに故障の影響が現れる．このとき，スキャン入力には，次のテストベクトルで要求される状態に相当する値を印加する．

スキャン設計による欠点を列挙すると次のようになる [2].

1)通常のFFをスキャンFFに変更することに伴い，ゲート数や回路面積が増大する．

2)少なくとも一本の外部入出力端子の付加が必要である．例えば，文献[22]のスキャン設計法では，4本の外部入出力端子の付加が必要である．図2.16の回路では，MCとSIの入力端子とSOの出力端子が必要であるが，SIとSOは元からある他の外部入力端子や外部出力端子と共有することで，一本の外部入出力端子の付加で済ませることができる．

3)1つのテストベクトルに対して，スキャンFFに等しい長さのスキャンシフト操作が必要となるため，テスト系列長が長くなる．

4)スキャンチェーンやスキャンFFにより遅延が大きくなり，元の回路の性能を低下させる．

このようにスキャン設計にはいくつかの欠点が存在するが，テスト生成が容易で高い故障検出率が得られるという利点のため，スキャン設計は，順序回路に対するテスト容易化設計法として現在最も広く用いられている．

2．7　まとめ

この章では，順序回路のモデル，故障モデル，テスト生成の概念と故障シミュレーションについて説明し，その後，順序回路のテスト生成法の一つ，時間展開法を紹介した．時間展開法は，順序回路のフィードバックループをなくした時間展開回路に対して，組合せ回路に対するテスト生成法と同様の方法を用いてテスト系列を生成する方法である．時間展開法を用いた場合でも，時間展開数の上限や多重故障，初期化の問題などにより，組合せ回路に対する場合と違い，回路内のすべての単一縮退故障に対してテスト系列を求めることは容易でない場合がある．

順序回路に対するテスト生成の複雑さを緩和するためのテスト容易化の一つの方法として，スキャン設計法を説明した．スキャン設計を行うことにより，順序回路のテスト生成の問題は組合せ回路のテスト生成の問題となり，比較的容易に高い故障検出率を得ることが可能になる．しかし，スキャン設計の欠点として，回路面積の増大，入出力端子の付加，テスト系列長の増大，回路性能の低下などが起こる．そのため，スキャン設計に代わる新しいテスト容易化設計法の開発や，スキャン設計の欠点を軽減するためのテスト法の開発が必要となっている．

第3章　短縮スキャンシフト法

短縮スキャンシフト法[54], [55]はスキャン回路に対するテスト系列を短くするための新しいテスト技法であり，通常のスキャン回路の構造以外に特別な回路構造を必要とせず，テスト系列生成を工夫するだけで実現できる．ここでは短縮スキャンシフトの基本的概念とその特徴について説明する．短縮スキャンシフト法をフルスキャン回路，パーシャルスキャン回路，リタイミング技法に応用した場合のテスト容易化設計法とテスト系列生成法について，それぞれ第4章，第5章，第6章で述べる．

3．1　基本概念

スキャン回路に対するテスト系列長が(2.1)式で表されることは，すでに述べた．スキャン回路のテスト系列長を短縮するためには，テストベクトル数かスキャンシフトを短くしなければならない．提案する短縮スキャンシフト法は，組合せ部分に対するテストベクトルが与えられた後に，テストベクトルを印加する前後に必要なスキャンシフトをできるだけ短くする方法である．一般に，ある故障を検出するために，常に全てのFFの制御と観測を行う必要はなく，一部のFFの制御と観測で十分である．もしテストベクトルを印加する際に，0でも1でもよいFFがあれば，そのFFを制御する必要はない．例えば図3.1(a)のように，テストベクトル印加前に必要な状態のFF_3とFF_4の値がドントケアである場合には，FF_1とFF_2だけを制御するためのスキャンシフトでよい．また，図3.1(b)のように，故障の影響が伝搬しないFFがあれば，そのFFを観測する必要はない．このような制御や観測の必要のないFFを効率よく見つけ，そのFFをスキャンシフトの対象から外すことによって，スキャンシフトの長さを短くする方法が短縮スキャンシフト法である．スキャンシフトの長さとは，

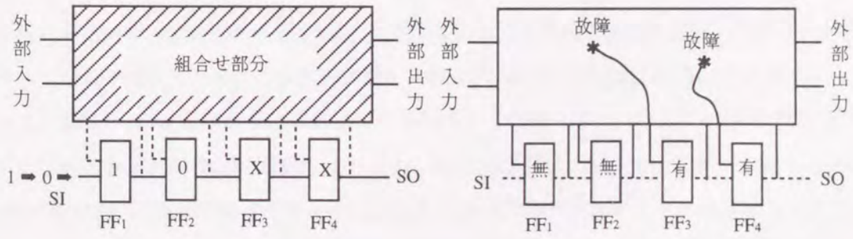


図3.1　設定すべき値と故障の影響の伝搬

スキャンチェーン上のFFをシフトするビット数で，長さkのスキャンシフトでは，スキャン入力側からk個のFFの値が，スキャン入力(SI)からの値によって設定され，スキャン出力側からk個のFFの値が，スキャン出力(SO)で観測される．このような短縮スキャンシフト法に対して，どのテストベクトルの印加前後にも，すべてのFFの制御と観測を行うようにスキャンシフト操作を行う従来の方法をフルスキャンシフト法とよぶ．(2.1)式で示したテスト系列長はフルスキャンシフト法によるものであるが，スキャンシフトの長さが常にFFと一致する．

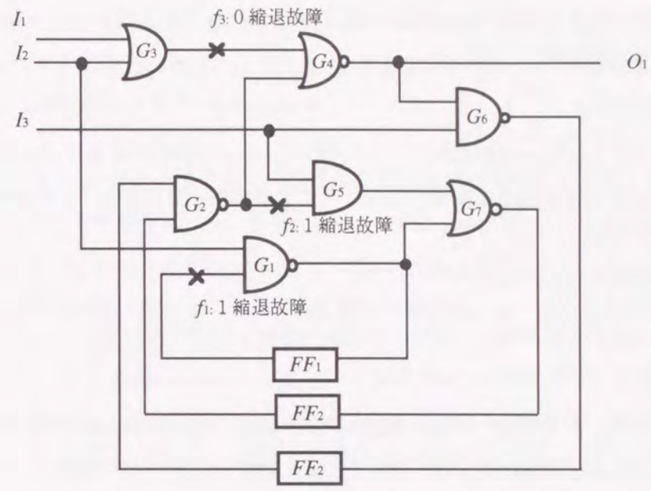
短縮スキャンシフト法を用いてテスト系列を短くするためには，制御や観測の必要のないFFの求め方とスキャンチェーン上のFFの配列が重要となる．1つのテストベクトルに対して，検出すべき故障と，スキャンチェーン上のFFの配列と，テストベクトル印加前の回路の状態が決まれば，スキャンシフトの長さを順に変化させて故障シミュレーションを行うことで，そのテストベクトルに対する最短のスキャンシフトの長さは一意に決まる．しかし2つのテストベクトルの間に必要なスキャンシフトの長さは，前後のテストベクトルに依存するため，テストベクトルを印加する順序の決め方が問題になる．スキャンシフト操作では，FFの制御と観測が同時に行われるため，たとえ観測のためのスキャンシフトの長さが短くても，次に印加するテストベクトルに対して必要なFFの制御のためのスキャンシフトの長さが長ければ，結果として必要なスキャンシフトの長さは長くなる．従って短縮スキャンシフト法における問題は，1)各テストベクトルが検出すべき故障，2)スキャンチェーン上のFFの配列，3)テストベクトルを印加する順序を，効率よく決めることである．なぜこれらの問題が重要であるかについて例を用いて説明する．

例3.1：図3.2の回路において3つの故障f_1，f_2，f_3を仮定する．故障f_1はFF_1の出力線の1縮退故障，f_2はG_2の分岐の枝の1縮退故障，f_3はG_3の0縮退故障である．これらの故障を検出するテストベクトルとして図3.2に示されるt_1，t_2が与えられたとする．故障f_1，f_2はそれぞれt_1，t_2のみによって検出され，故障f_3はその両方のテストベクトルで検出される．テストベクトルt_1，t_2が検出すべき故障を変化させたときに，制御や観測の必要となるFFは表3.1に示される．例えば，t_1が検出すべき故障としてf_1，f_3を選んだときは，FF_1とFF_2とFF_3の制御が必要であるが，f_1だけを検出すればよいのであれば，FF_1の制御だけでよい．一方，観測すべきFFについてはどちらの場合もFF_1を観測しなければならない．テストベクトルt_2では，対象故障の違いによって，制御と観測すべきFFは同じである．このことから，t_2によって故障f_3を検出するようにしたとき，制御と観測の必要なFFの数が少なくなることが分かる．以下では，f_3をt_2で検出するとして議論する．

次に，スキャンチェーン上のFFの配列について考える．6通りのFFの配列の組合せの中から最適なものを選ぶ．テストベクトルt_2では，FF_1, FF_2, FF_3の制御と，FF_2の観測が必要である．制御に関しては，FFの配列に依存せず，長さ3のスキャンシフトが必要であるが，観測に関しては，FF_2をスキャン出力に近いところに配列することによって，長さ1のスキャンシフトで済む．従って，スキャン入力側からFF_1 - FF_3 - FF_2のような配列した時に，スキャンシフトの長さはより短くなる．

テストベクトルt_1に関しては，FF_1の制御と観測の両方を行わなければならないため，どのような配列であってもスキャンシフトの長さの全体は変わらない．

次に，FF_1 - FF_3 - FF_2の配列の下で，テストベクトル印加の順序とテスト系列長について調べる．



テストベクトル
t1(I1, I2, I3, FF1, FF2, FF3) =(0, 1, 0, 0, 1, 1)
t2(I1, I2, I3, FF1, FF2, FF3) =(0, 1, 1 , 1, 1, 1)

図3.2　順序回路例とテストベクトル

表3.1　対象故障とFFの制御・観測

テスト ベクトル	対象 故障	設定すべき値 (FF1, FF2, FF3)	観測すべきFF (FF1, FF2, FF3)
t1	f1	(0, X, X)	(⊕, - , -)
	f1, f3	(0, 1, 1)	(⊕, - , -)
t2	f2	(1, 1, 1)	(-, ⊕, -)
	f2, f3	(1, 1, 1)	(-, ⊕, -)

X：ドントケア　⊕：故障の影響の伝搬有

もし，t_1 - t_2の順でテストベクトルを印加すると，t_1 - t_2の間のスキャンシフト操作では，FF_1の観測とFF_1, FF_2, FF_3の制御を行うために，長さ3のスキャンシフトが必要となる．一方，t_2 - t_1の順でテストベクトルを印加すると，FF_2の観測とFF_1の制御を行うために，スキャンシフトの長さは1ですむ．

以上のように，対象故障，FFの配列，テストベクトルの印加の順序を変化させることによって，スキャンシフトの長さが変化し，結果として，テスト系列長が変化することが分かる．

3．2　関連する過去の研究

スキャン回路のテスト系列短縮を目指した過去の研究について紹介する．

組合せ回路に対するテストベクトル数を減らす研究として[67]-[69]がある．テストベクトル数を減らすことは，スキャン回路のテスト系列短縮に寄与するが，多くのFFを含む回路に対しては，スキャンシフトの長さが依然長いままで，さらに短縮が可能である．以降では，スキャンシフトの長さの短縮を目指した研究について紹介する．それらは，大別すると，次のように分類できる．

a) 回路構造の変更を伴う手法

b) 回路構造の変更を伴わない手法

Chenら[56]は，スキャンシフト操作で制御と観測されるFFの数が1からn(FF数)まで変化するような特殊なスキャンチェーンを構成し，テストベクトル生成の際には，対象故障がスキャン入力側のk (k = 1,..., n-1)個のFFの値の設定で検出できるかを調べ，検出できなければk+1個のFFの値を設定するということを順次繰り返す方法を提案している．Morleyら[57]は，スキャンFFを二つのグループに分け，スキャンシフトするグループが選択できるような回路構造を提案している．FujiwaraとYamamoto[58]は，スキャンFFへの入力値の排他的論理和が外部で観測できるような回路構造を構成する手法を提案している．以上の三手法は，特別な構造のスキャン回路を構成する手法として分類できる．

GuptaとBreuer[59]は，スキャンチェーン内のスキャンレジスタ(スキャンFFの一群)の順序を考えることによって，テスト系列短縮を実現している．またNarayananら[60]は，文献[59]と同様の手法を多重スキャンチェーンに応用した手法を報告している．これらの手法は，スキャンFFの順序を指定する以外は，どのようなスキャン回路であっても適用可能である．

LeeとSaluja[61],[62]は，順序回路に対するテスト系列生成アルゴリズムを改良し，必要に応じてスキャンシフト操作を行うようなテスト系列の生成法を提案している．Linら[63],[64]は，順序回路と組合せ回路に対する二つのテスト生成アルゴリズムを用い，スキャンシフト数を削減している．ただし，順序回路に対するテスト系列生成アルゴリズムを用いた場合には，多くのFFを含む回路に対してよい結果が得られないことが多い．Higuchiら[65]は，BDDを用いて短いテスト系列を生成している．上記の文献[59]-[65]の手法は，テスト系列生成において何らかの工夫をしており，スキャンシフト操作が可能であれば回路の構造に依存しない手法である．

3．3　制御と観測に必要なスキャンシフト

[定義3.1]テストベクトルt_iを印加する前に必要な，FFの制御のためのスキャンシフトの長さを$NSC(t_i)$と表し，スキャン制御数という．また，テストベクトルt_iを印加した後に必要な，FFの観測のためのスキャンシフトの長さを$NSO(t_i)$と表し，スキャン観測数という．スキャンシフト操作では，FFの制御と観測が同時に行われるので，特に制御や観測を区別しない場合のスキャンシフトの長さをスキャンシフト数という．

スキャンシフト操作では，FFの制御と観測が同時に行えるため，短縮スキャンシフト法によるテスト系列生成の際，2つのテストベクトルt_iとt_{i+1}の間に必要なスキャンシフト数を，スキャン観測数とスキャン制御数の大きい方と決める．従って，テスト集合 $T = \{ t_1, t_2,...,t_v \}$が与えられ，$t_1$, t_2,...,t_vの順に印加するときの，短縮スキャンシフトによるテスト系列長は次式で与えられる．

$$L_r = \sum_{i=1}^{v-1} \max\{NSO(t_i), NSC(t_{i+1})\} + NSC(t_1) + NSO(t_v) + v \qquad (3.1)$$

例3.2：5つのスキャンFFを含む回路に対して，表3.2に示されるような3つのテストベクトルが生成されたと仮定する．テストベクトルt_1, t_2, t_3は，それぞれ故障f_1, f_2, f_3を検出すると仮定する．また，故障f_1, f_2, f_3の影響がどのFFに伝搬するかを調べた結果が表3.3で，例えばt_1を印加したときには，故障f_1の影響はFF_4に伝搬する．スキャンチェーン上のFFが，スキャン入力から順にFF_1 - FF_2 - FF_3 - FF_4 - FF_5のように配列されていると仮定したとき，各テストベクトルに対するスキャン制御数とスキャン観測数は表3.4のように求められる．テストベクトルt_1に対しては，FF_1とFF_2を制御しなければならないので，スキャン制御数は2，また故障f_1の影響が伝搬するFF_4を観測しなければならないので，スキャン観測数は2となる．テストベクトルをt_1, t_2, t_3の順に印加すると仮定したときの，短縮スキャンシフト法によるテスト系列長は(3.1)式より15となる．

フルスキャンシフト法を仮定したときには，(2.1)式より，テスト系列長は23となる．

3．4　対象故障

制御や観測の必要なFFを求める際に，与えられたテスト集合の各テストベクトルでどの故障を検出すべきかを決定する方法について述べる．あるテストベクトルが検出すべき故障のことを，こ

こでは，そのテストベクトルの対象故障という．1つのテストベクトルの対象故障の数が少ない方が，制御や観測の必要なFFの数が少なくなるのは明らかで，結果としてスキャンシフト数も少なくなる．テストベクトルから得られたテスト系列で未検出故障が残らないためには，全ての故障がテスト集合のうちの少なくとも1つのテストベクトルの対象故障とならなければならないが，複数のテストベクトルで検出される故障を，制御や観測の必要なFFが少なくなるように，各テストベクトルに割り当てることは非常に難しい．そこで，提案方法では，次のような予想を立て，対象故障を減らす方法を考える．

[予想3.1]多くの故障は複数のテストベクトルで検出され，それらの故障は対象故障に選ばなくとも，最終的に得られたテスト系列で検出される可能性が高い．

１つの故障が，与えられたテスト集合の中のいくつのテストベクトルによって検出されるかを調べた実験の結果を表3.5に示す．表中の各列は，全故障数，全テストベクトル数，1つのテストベクトルでしか検出されない故障数，2または3，または4個のテストベクトルによって検出される故障数の和，5個以上のテストベクトルによって検出される故障数の和を表す．また括弧内には故障数の全故障数に対する割合を%で表す．

予想3.1に基づき，少ない数のテストベクトルでしか検出されない故障を対象故障に選び，制御や

表3.2　テストベクトル

テストベクトル	対象故障	(I1, I2, FF1, FF2, FF3, FF4, FF5)
t1	f1	(1, 0, 1, 1, X, X, X)
t2	f2	(1, 1, 0, 0, 1, X, X)
t3	f3	(0, 0, 1, 0, 0, 1, X)

X：ドントケア（制御の必要無）

表3.3　観測すべきFF

テストベクトル	観測すべきFF (FF1, FF2, FF3, FF4, FF5)
t1	(-, -, -, ⊕, -)
t2	(-, ⊕, -, -, -)
t3	(-, -, ⊕, -, -)

⊕：故障の影響の伝搬有

表3.4　NSCとNSO

	NSC	NSO
t1	2	2
t2	3	4
t3	4	3

観測の必要なFFを求める．提案方法では，テスト集合の中の唯一つのテストベクトルでしか検出されない故障を対象故障に選ぶ．テスト集合の中の唯一つのテストベクトルでしか検出されない故障を必須故障とよぶ[67]．表3.5に示されるように，必須故障の割合は一般的には少ないので，必須故障だけを対象にすることで，制御や観測の必要なFF数は少なくなるが，必須故障以外の故障のいくつかが，得られたテスト系列では未検出となる可能性がある．未検出故障が残った時の処理については第4章で詳しく述べる．

3．5　制御や観測の必要なフリップフロップの求め方

テスト集合と各テストベクトルの対象故障が与えられたときに，制御や観測の必要なFFは，故障シミュレーションを行うことによって求められる．まずそれらを計算するときの問題点を指摘し，その後，提案方法について述べる．問題点は，

1)FFのあらゆる組合せに対して，制御の必要なFFを求める計算は非常に莫大な計算量となること，

2)制御や観測の必要なFFが常に一意に決まるとは限らないこと，

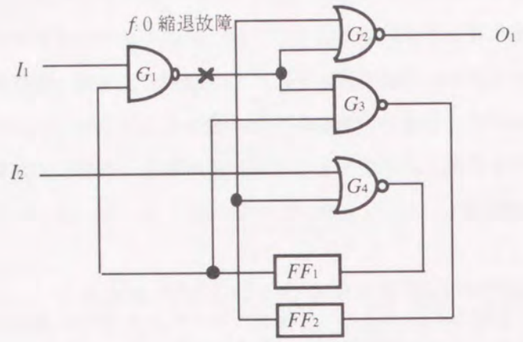
3)計算に用いる信号値の種類によって結果が異なること，

である．

FF数がnであるとき，FFの組合せは2^n通りあり，nが少しでも大きくなると，これらをすべて計算するには非常に莫大な計算時間が必要で，実用的には不可能である．従って，問題点1)の解決には，FFを選択して計算することが必要となる．問題点2)は，テストベクトルに対して対象故障が決まったときでも，制御や観測の必要なFFが一意に決まるとは限らないことで，つまり，ある故障が

表3.5　故障と検出されるテストベクトル数

回路	全ベクトル	全故障数	1ベクトルで検出される故障数	2-4ベクトルで検出される故障数の和	5ベクトル以上で検出される故障数の和
s208	28	215	46(21.4%)	68(31.6%)	101(47.0%)
s344	18	342	77(22.5%)	110(32.2%)	155(45.3%)
s382	27	399	74(18.9%)	156(39.1%)	169(42.0%)
s1196	126	1242	238(19.2%)	222(17.8%)	782(63.0%)



テストベクトル
(I1, I2, FF1, FF2)
t　= (0, 1, 1, 0)
t1 = (0, 1, X, 0)
t2 = (0, 1, 1, X)

図3.3　回路例とテストベクトル

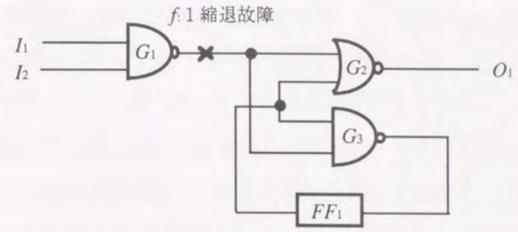
表3.6　テストベクトルと信号線の値

	I1	FF1	FF2	G1	G2	G3
t	0/0	1/1	0/0	1/0	0/1	0/1
t1	0/0	X/X	0/0	1/0	0/1	X/1
t2	0/0	1/1	X/X	1/0	0/X	0/1

0/1: 正常値／故障値

検出されたとしても，制御するFFによって故障の影響の伝搬する経路が異なり，結果として観測すべきFFが異なる場合があることを意味する．例えば図3.3に示されるように，ゲートG_1の0縮退故障fと，それを検出するテストベクトルとして$t = (I_1, I_2, FF_1, FF_2) = (0, 1, 1, 0)$が得られたと仮定する．故障$f$はテストベクトル$t_1 = (0, 1, X, 0)$でも，$t_2 = (0, 1, 1, X)$でも検出され，それぞれに対する各ゲートの出力信号線の値は表3.6に示される．この表よりt_1では故障の影響が外部出力に伝搬し，t_2ではFF_2に伝搬することが分かる．すなわち，t_1，t_2のように制御するFFが異なると，観測すべきFFも異なる場合がある．

問題点3)は，故障シミュレーションを行う場合に，制御の必要性を調べたいFFの値を，元の値の反転した値を用いるか，またはドントケアXを用いるかによって結果が異なることを意味する．例えば，図3.4の回路でゲートG_1の1縮退故障fと，それを検出するテストベクトル$t = (I_1, I_2, FF_1) = (1, 1, 0)$が得られたと仮定する．故障$f$を検出するために，$FF_1$の値0が必要かどうか，つまり，$FF_1$は制御の必要があるかどうかを調べる．元のテストベクトルtから2つのテストベクトルt_1とt_2を



テストベクトル
(I1, I2, FF1)
t　=(　1,　1,　0　)
t1=(　1,　1,　X　)
t2=(　1,　1,　1　)

図3.4　回路例とテストベクトル

表3.7　テストベクトルと信号線の値

	I1	I2	FF1	G1	G2	G3
t	1/1	1/1	0/0	0/1	1/0	1/1
t1	1/1	1/1	X/X	0/1	X/0	1/X
t2	1/1	1/1	1/1	0/1	0/0	1/0

0/1: 正常値／故障値

つくる．テストベクトルt_1では，FF_1の値をXにし，t_2では0を反転させた1を割り当てる．t_1，t_2に対して故障シミュレーションした時の各ゲートの出力線の値は表3.7に示される．この表より，$t_1 = (1, 1, X)$は故障fを検出しないが，$t_2=(1, 1, 1)$はfを検出することが分かる．tでもt_2でもfが検出されるということは，FF_1は0でも1でもよく，制御の必要のないことを示す．ところがt_1における故障シミュレーションでは，fが検出されないため，FF_1は制御しなければならないという結果を得ることになる．

以上3つの問題点を考慮の上，必須故障を対象に，極大圧縮法[68]を用いて制御や観測の必要なFFを調べる．極大圧縮法では，元のテストベクトルから1つのFFの値を反転させたテストベクトルを作り，対象故障に対して故障シミュレーションを行う．もし，FFの値を反転したことによって，1つでも未検出故障が残れば，元のテストベクトルで決められた0または1の値にそのFFを必ず制御しなければならないことが分かる．また対象故障がすべて検出されたときには，故障の影響の伝搬するFFを見つけ，そのFFを観測しなければならないことが分かる．この方法では，1つのテストベクトルに対し，n(FF数)種類の異なるテストベクトルに対する計算量が必要で，これは2^nの

計算量に較べて非常に少ない量である．また，Xを用いた場合の問題も生じない．ただし，すべてのFFの組合せを尽くしたわけではなく，制御の必要のないFFの値がすべて同時に反転したテストベクトルでは，未検出故障が残る可能性がある．計算アルゴリズムを図3.5に示し，計算の例を次に示す．

例3.3：図3.6の回路に対し，テストベクトル$t = (I_1, FF_1, FF_2, FF_3) = (1, 1, 1, 0)$とその必須故障としてゲート$G_3$の出力の0縮退故障$f$が与えられたとする．$t$に対し，各FFの値を反転させた$t_1$，$t_2$，$t_3$のテストベクトルをつくり，故障シミュレーションを行う．表3.8はシミュレーションでの各ゲートの値を示す．表3.8より，t_1は故障fを検出しないので，FF_1は制御の必要があると分かる．しかし，この方法では，FF_2とFF_3は制御の必要がないという結果を得るが，それらのFFの値が必ずしも任意であることを意味しない．即ちテストベクトル$t_{23} = (I_1, FF_1, FF_2, FF_3) = (1, 1, 0, 1)$に対しては故障$f$は未検出となることが，表3.8より分かる．

3．6　テストベクトル生成

短縮スキャンシフトにおいて，テスト系列を短くするためは，次のような性質を持つテストベクトルを生成することが望ましい．

```
Procedure: Maximal compaction
/* T is a set of test vectors */
for ( every test vector t_i in T )
{
    for ( every fault f_j which is an essentail fault for t_i )
    {
        for ( k = 1 to n ) /* n is the number of FFs */
        {
            Invert the value of only FF_j ;
            if ( fault f_j is not combinationally detected )
                FF_j must be controlled ;
            else if ( the effect of f_j is propagated to a FF )
                The FF must be observed ;
        }
    }
}
```

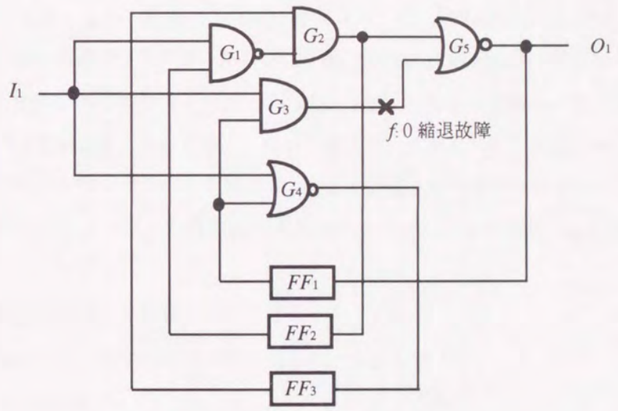
図3.5　制御・観測の必要なFFを見つけるアルゴリズム

1)全体のテストベクトル数が少ない

2)スキャン制御数が小さい

3)スキャン観測数が小さい

テストベクトル生成の時に，テストベクトル中にできるだけ多くのドントケアを含むようにすれば，スキャン制御数が小さくなると考えられる．また，できるだけ故障の影響を外部出力へ伝搬させるようにすれば，スキャン観測数が小さくなると考えられる．スキャン制御数やスキャン観測数を小さくするための1つの方法は，1つのテストベクトルで検出する故障数を少なくすることである．ところが，全体のテストベクトル数を小さくするためには，1つのテストベクトルが検出する故障をできるだけ多くする必要がある．つまり，全体のテストベクトル数を小さくすることと，各



テストベクトル
t (I1, FF1, FF2, FF3) =(1, 1, 1, 0)

図3.6　回路例とテストベクトル

表3.8　テストベクトルと信号線の値

	I1	FF1	FF2	FF3	G1	G2	G3	G5
t	1/1	1/1	1/1	0/0	0/0	0/0	1/0	0/1
t1	1/1	0/0	1/1	0/0	0/0	0/0	0/0	0/0
t2	1/1	1/1	0/0	0/0	1/1	0/0	1/0	0/1
t3	1/1	1/1	1/1	1/1	0/0	0/0	1/0	0/1
t23	1/1	1/1	0/0	1/1	1/1	1/1	1/0	0/0

0/1: 正常値／故障値

テストベクトルのスキャン制御数やスキャン観測数を小さくすることを，テストベクトル生成において両立することは難しい．そこで，提案手法では，文献[69]の手法を用い，全体のテストベクトル数が小さくなるように，テストベクトルを生成する．

第4章　フルスキャン回路に対する短縮スキャンシフト法

フルスキャン回路に対して短縮スキャンシフト法を応用したときの，テスト容易化設計法とテスト系列生成法[54], [55]について述べる．本章では特に提案手法の特色である，スキャンチェーンの構成法と，テスト系列を生成する際のテストベクトルの順序の決め方について詳しく説明し，その後ベンチマーク回路に対する実験結果を示す．ここでの目的は，短いテスト系列を生成することと，組合せ部分のテストベクトルで得ることのできる故障検出率を低下させないことである．

4．1　スキャンチェーンの構成

スキャン制御数やスキャン観測数は，スキャンチェーン上のFFの配列にも依存する．もし，あるテストベクトルに対して制御の必要なFFがスキャン入力から遠ければ，そのFFの値を制御するためのスキャンシフト数は大きくなり，また，観測の必要なFFがスキャン出力から遠ければ，そのFFの値を観測するためのスキャンシフト数は大きくなる．従って，頻繁に制御の必要なFFがスキャン入力から遠くに存在する場合や，頻繁に観測の必要なFFがスキャン出力から遠くに存在する場合には，故障検出のために必要なスキャンシフト数が大きくなると考えられる．そこで，スキャンシフト数を小さくするために，各FFについて，いくつのテストベクトルに対して，制御や観測をしなければならないかを調べ，その数の違いによってスキャンチェーン上のFFの配列を決定する．FFの制御や観測の必要性を次の尺度を用いて計算する．

$VC(FF_i)$：FF_iを制御することを必要とするテストベクトルの数

$VO(FF_i)$：FF_iを観測することを必要とするテストベクトルの数

$$W(FF_i) = VC(FF_i) - VO(FF_i) \qquad (4.1)$$

極大圧縮法によって，制御や観測の必要なFFが求まれば，VC, VO, Wは簡単に計算することができる．Wが大きいFFは制御の必要性が高く，Wが小さいFFは観測の必要性が高いことを意味するので，Wの大きい順にスキャン入力側からFFを配列し，スキャンチェーンを構成する．

4．2　テストベクトルの順序

ここでは，各テストベクトルに対するスキャン制御数とスキャン観測数，スキャンチェーン上のFFの配列が決まった後の，テストベクトルを印加する順序の決め方について提案した方法を説明する．スキャンシフト数は直前に印加したテストベクトルのスキャン観測数と，直後に印加するテストベクトルのスキャン制御数の大きい方とするため，テストベクトルを印加する順序によって，テスト系列長は変化する．ここで考えるべき問題は，v個のテストベクトル t_1～t_v とそれらに対するスキャン制御数 $NSC(t_i)$，スキャン観測数 $NSO(t_i)$ が与えられたとき，(3.1)式で表されるテスト系列長 L_t ができるだけ短くなるようなテストベクトルの順序を見つけることである．

(3.1)式の $\sum_{i=1}^{v-1}\max\{NSO(t_i), NSC(t_{i+1})\}$ が最小であるための十分条件を与える次の補題が示される．

[補題4.1] $i = 1, \ldots v\text{-}1$ に対して，$NSO(t_i) = NSC(t_{i+1})$ ならば，$L_s = \sum_{i=1}^{v-1}\max\{NSO(t_i), NSC(t_{i+1})\}$ は最小である．

（証明）$D_i = \begin{cases} NSO(t_i) - NSC(t_{i+1}) & \text{if } NSO(t_i) > NSC(t_{i+1}) \\ 0 & \text{otherwise} \end{cases}$

とすると，

$$L_s = \sum_{i=1}^{v-1}\max\{NSO(t_i), NSC(t_{i+1})\} = \sum_{i=1}^{v-1}\{NSC(t_{i+1}) + D_i\}$$

$\sum_{i=1}^{v-1} NSC(t_{i+1})$ はテストベクトルの順序に拘わらず一定であるので，$\sum_{i=1}^{v-1} D_i$ が最小のとき L_s は最小となる．$i = 1, \ldots v\text{-}1$ に対して，$NSO(t_i) = NSC(t_{i+1})$ のとき，$\sum_{i=1}^{v-1} D_i$ は0となり最小であるので，L_s は最小となる．（証明終）

（3.1）式において，$\sum_{i=1}^{v-1}\max\{NSO(t_i), NSC(t_{i+1})\}$ と $NSC(t_1) + NSO(t_v)$ は独立であるので，$\sum_{i=1}^{v-1}\max\{NSO(t_i), NSC(t_{i+1})\}$ が最小のとき，テスト系列長 L_r が最小となるとは限らないが，$\sum_{i=1}^{v-1}\max\{NSO(t_i), NSC(t_{i+1})\}$ は $NSC(t_1) + NSO(t_v)$ より大きいと考えられるため，ここでは，$\sum_{i=1}^{v-1}\max\{NSO(t_i), NSC(t_{i+1})\}$ が小さくなるように，補題4.1に基づき，$NSO(t_i)$ と $NSC(t_{i+1})$ の差が小さくなるようにテストベクトルの順序を決める．スキャンシフト操作では，FFの制御と観測が同時に行われるため，$NSO(t_i)$ と $NSC(t_{i+1})$ の差が小さいことは，FFの制御と観測の両方に必要なスキャンシフトが行われたことで，逆に $NSO(t_i)$ と $NSC(t_{i+1})$ の差が大きいことは，制御のみ，または観測のみのためにスキャンシフトが行われたことを意味する．テストベクトルの順序を決める際には，計算時間を少なくするために，バックトラックを行わず，1番目から順に選ぶ方法を採る．1番目のテストベクトルとしては，*NSC* が最小のテストベクトルを選ぶ．テストベクトルの順序を決める規則は，次のようにまとめることができる．

規則1）*NSC* が最小のテストベクトルを1番目に選ぶ．候補が複数存在する場合は，計算上先に見つけたものを選ぶ．

規則2a）i 番目までのテストベクトルが決定し，i+1番目のテストベクトルを選ぶとき，$|NSO(t_i) - NSC(t)|$ が最小のテストベクトル t を選ぶ．候補が複数存在する場合は，計算上先に見つけたものを選ぶ．

表4.1　NSCとNSO

	NSC	NSO
t1	3	3
t2	5	1
t3	2	4
t4	4	2

例4.1：4つのテストベクトルとそれに対するスキャン制御数とスキャン観測数が表4.1のように与えられたとする．規則1と規則2aに従い，t_3 - t_4 - t_1 - t_2 のようにテストベクトルの順序が決まり，総スキャンシフト数は15となる．

4．3　テスト系列生成の繰り返し

制御や観測の必要なFFを調べる際の対象故障として必須故障を選ぶことは，計算時間の節約や，スキャンシフト数の減少に有効であったが，反面，必須故障以外の故障が，生成したテスト系列に対して未検出となる可能性がある．そこで，生成したテスト系列に対して未検出故障が生じたときには，その故障を必ず検出するようにスキャンシフト数を増やし，テスト系列の生成を繰り返す方法を用いる．フルスキャン回路では，すべてのFFをスキャンシフトすることによって，元のテストベクトルが組合せ回路的に検出する故障は必ず検出されることを保証される．必須故障に含まれない故障が未検出故障とし

```
Procedure: Test sequence generation
/* T = { t1, t2, ... tv } is a set of test vectors. */
/* Fi is a set of essentail faults for ti. */
AGAIN:
for ( i = 1 to v )
    Calculate NSC(ti) and NSO(ti) for Fi ;
Select a test vector as the first test vector such that NSC(t) is minimum;
for ( i = 2 to v )
    Select a test vector as the i-th test vector according to the Rule_2a;
if ( there is no undeteced faults by the generated test sequence )
    exit;
for ( every undetected fault f )
{
    Find a test vector ti which combinationally detects the fault;
    Fi = Fi + { f };
}
Go to AGAIN;
```

図4.1　テスト系列生成アルゴリズム

て残る原因は，

1)スキャン制御数が小さいために，必要な状態が設定されなかった，

2)スキャン観測数が小さいために，FFに伝搬していた故障の影響を観測することができなかった，

のいずれか一方，または両方である．従って，未検出故障が残ったときには，それを検出するテストベクトルを見つけ，それに対するスキャン制御数やスキャン観測数を大きくすれば，その未検出故障は必ず検出される．テスト系列生成をやり直すときには，テストベクトルの対象故障に未検出故障を含めてスキャン制御数とスキャン観測数を再計算する．そして，再計算されたスキャン制御数とスキャン観測数に対して，もう一度，規則1と規則2aに従って，テストベクトルの順序を決め直す．図4.1にアルゴリズムを示す．このアルゴリズムは，与えられたテストベクトルが達成する故障検出率と同じ故障検出率が必ず得られることを保証する．

4．4　状態遷移を利用したテスト系列生成

テスト系列中のスキャンシフト数を決定するときに，そのときの回路の状態を考慮することによって，さらにスキャンシフト数を小さくする方法を述べる．もし，スキャンシフトによって制御しなければならないFFの値が，テストベクトルを印加した後の状態とすべて一致したなら，スキャンシフトを行う必要がなくなる．テストベクトル中の制御しなければならないFFの値と，回路の状態を比較することによって，回路の状態と異なるFFだけを制御すれば，スキャンシフト数をさらに減らすことができる．ただし，回路の状態もスキャンシフト操作に同期して変化していくため，状態遷移を考慮したときのスキャン制御数は，印加するテストベクトルを決める度に，動的に計算する必要がある．状態遷移を考慮したときのスキャン制御数をNSC_{st}と定義し，それを次のように計算する．

［NSC_{st}の計算］

1)回路の状態を表す状態ベクトル $st(FF_1, FF_2, ..., FF_n)$ をつくる．スキャン入力に近い順に$FF_1, FF_2, ..., FF_n$とする．

2)テストベクトル t_i に対し，FFの値だけを含む，ベクトル $t_i^s(FF_1, FF_2, ..., FF_n)$をつくる．ただし，制御の必要のないFFの値はXとする．

3)k を 0 から n-1 まで変化させて以下の操作を行う．

3.1) st の値を右にkビットシフトさせたベクトル st_k をつくる．右に1ビットシフトさせるとは，FF_iの値をFF_{i-1}の値にすることである．

3.2) st_k と t_i^sのFF_{k+1}からFF_nまでのXでない各ビットを比較し，それらがすべて一致したなら，$NSC_{st}(t_i)$をkとし，処理を終了する．

テストベクトルの順序を決める規則2aを次のように変更し，テスト系列を生成する．

表4.2　テストベクトルとNSC

	(FF_1, FF_2, FF_3, FF_4)	NSC
$t1^s$	(1, 1, 0, X)	3
$t2^s$	(1, 0, 0, 1)	4

表4.3　状態とビットシフト

	(FF_1, FF_2, FF_3, FF_4)
st	(0, 0, 1, 0)
st1	(-, 0, 0, 1)
st2	(-, -, 0, 0)
st3	(-, -, -, 0)

規則2b）j 番目までのテストベクトルが決定し，j+1番目のテストベクトルを選ぶとき，$NSC_{st}(t) \geq NSO(t_j)$ かつ $NSC_{st}(t)$が最小のテストベクトル t を選ぶ．もし$NSC_{st}(t) \geq NSO(t_j)$を満たすテストベクトル$t$が存在しないときには，$NSC_{st}(t)$が最小のテストベクトル t を選ぶ．複数の候補が存在するときには，計算上先に見つけたものを選ぶ．

例4.2：テストベクトル t_j を印加した後の状態が $st(FF_1, FF_2, FF_3, FF_4) = (0, 0, 1, 0)$で，$NSO(t_j)$が1と仮定する．このとき，次に印加すべきテストベクトルとして，2つのt_1, t_2のいずれかから選ぶことを考える．テストベクトルt_1, t_2とそれに対するNSCが表4.2のように与えられたと仮定する．またスキャンチェーン上のFFはスキャン入力側からFF_1, FF_2, FF_3, FF_4の順に配列されていると仮定する．規則2aに従えば，t_1が選ばれ，そのときのスキャンシフト数は3となる．スキャンシフト操作を行う度に，状態ベクトル st は表4.3のように1ビットづつシフトしていく．ここで'-'で表されているFFは，スキャン入力からの値によって決まるため，考慮する必要がない．スキャンシフト操作により1ビットシフトした状態で，スキャン入力からの制御によってFF_1を1にすることで，t_2に必要な状態(1, 0, 0, 1)が得られる．即ち$NSC_{st}(t_2)$は1となる．同様に，$NSC_{st}(t_1)$は2となる．規則2bに従い，テストベクトルt_2を選ぶと，スキャンシフト数は1で，状態遷移を考慮しない場合より，スキャンシフト数は小さくなる．

4．5　実験結果

提案した方法をC言語によりプログラム化し，ISCAS'89ベンチマーク回路[52]に対しSun-SS/Classicワークステーション上で実験した．表4.4に実験結果を示す．"circuit"は回路名を表し，"PI"，"FF"，"V"，"R"はそれぞれ，外部入力数，FF数，テストベクトル数，検出不能故障数を表す．テストベ

表4.4　実験結果

Circuit	PI	FF	V	R	Lf	L1	L2	L1 / Lf (%)	L2/ Lf (%)	CPU1 (sec)	CPU2 (sec)	L[61]	L[65]
s208	11	8	32	3	296	224	171	75.7	57.8	6.9	4.4	NA	182
s298	3	14	26	0	404	308	194	76.2	48.0	5.1	7.8	376	222
s344	9	15	19	0	319	241	188	75.5	58.9	10.7	6.9	166	85
s349	9	15	17	2	287	223	161	77.7	56.1	8.1	6.8	125	86
s382	3	21	27	0	615	361	339	58.7	55.1	22.1	36.6	680	356
s386	7	6	68	0	482	462	272	95.9	56.4	13.5	11.6	376	264
s420	19	16	49	5	849	664	550	78.2	64.8	32.2	22.4	737	610
s444	3	21	27	14	615	346	285	56.3	46.3	21.9	26.2	788	287
s526	3	21	54	1	1209	789	688	65.3	56.9	66.4	67.0	1551	NA
s641	35	19	28	0	579	377	260	65.1	44.9	15.5	13.8	NA	NA
s820	18	5	102	0	617	602	295	97.6	47.8	53.6	72.2	617	440
s832	18	5	99	15	599	592	264	98.8	43.6	57.8	60.5	589	465
s838	35	32	78	10	2606	2300	2193	88.3	84.2	193	147	3263	NA
s953	16	29	82	0	2489	580	252	23.0	10.1	84.1	51.2	924	370
s1196	14	18	127	0	2431	664	506	27.3	20.8	149	141	465	197
s1238	14	18	137	69	2621	651	488	24.8	18.6	156	164	448	203
s1423	17	74	34	16	2624	1935	1827	73.7	69.6	462	491	3222	NA
s1488	8	6	108	2	762	757	292	99.3	38.3	102	93.8	641	580
s1494	8	6	107	13	755	751	285	99.5	37.7	103	95.0	582	564
s5378	35	179	109	39	19799	16175	15937	81.7	81.6	6203	4695	NA	NA

クトルは文献[69]によるテスト圧縮技法を用いて生成したもので，検出不能故障は，そのテストベクトル生成のときに識別されたものである．"Lf"はフルスキャンシフトの場合のテスト系列長で，式(2.1)により計算したもの，"L1"，"L2"は短縮スキャンシフト法によるテスト系列長で，状態遷移を利用しない場合と，状態遷移を利用した場合を表す．"L1/Lf"と"L2/Lf"はそれぞれ"L1"，"L2"の"Lf"に対する割合を表す．"CPU1"，"CPU2"は"L1"，"L2"に対応するテスト系列を求めるための計算時間を表す．ただし，テストベクトル生成のための計算時間は含まない．"L[61]"は，LeeとSaluja[61]による順序回路のテスト生成方法を応用した方法によるフルスキャン回路のテスト系列長の結果であり，"L[65]"は，Higuchiら[65]によるBDDを用いた方法でのフルスキャン回路のテスト系列長の結果である．計算機の使用メモリ量に関しては，s5378に対して，L1を得るために5.19 Mbyte，L2を得るために5.38 Mbyte 必要であった．

実験結果より，短縮スキャンシフト法によりテスト系列長を短くできることが示された．また状態遷移を利用することによって，さらに短くできることが示された．特にs386やs820やs1488のようなFF数の少ない回路において，状態遷移の利用が大変有効であった．また，状態遷移を利用したときでも，計算

時間とメモリの使用量はあまり変わらなかった．比較的良い結果を得ている2つの方法[61]，[65]と比べたときでも，多くの回路に対して，短縮スキャンシフト法がより短いテスト系列を得ることが示された．

4．6　まとめ

この章では，短縮スキャンシフト法をフルスキャン回路に応用したときの，スキャンチェーンの構成法とテスト系列生成法，実験結果について述べた．ここで対象とする回路は，フルスキャン回路であればどのような実装方式でもかまわず，スキャンシフト操作が可能であればよい．FFを配列しスキャンチェーンを構成することは，短縮スキャンシフト法をさらに有効にし，より短いテスト系列を得るために導入したのであるが，回路設計時のレイアウト上の制約により，実際にはFFの配列を自由に決定できない場合もある．しかしそのような場合であっても，短縮スキャンシフト法を適用することは可能で，そのことによって，フルスキャンシフト法よりも短いテスト系列を得ることが可能であると考えられる．現実的には，一部のFFのみが自由に配列可能である，または，ある制約の下で配列可能である状況を考えた短縮スキャンシフト法の開発が求められるが，このような場合にも，本論文で提案した方法，または同様の方法が適用可能であると考える．

実験で用いたテストベクトルは，テストベクトル数を少なくするためのテスト圧縮技法を用いて生成したが，そのことが短いテスト系列を得ることに貢献している．しかし，短縮スキャンシフト法は，他のどのような方法で生成されたテストベクトルにも有効で，テストベクトルの生成について特別な方法を要求しない．最終的に得られたテスト系列で達成できる故障検出率は，元のテストベクトルで達成できる組合せ部分での故障検出率と同じである．

第５章　パーシャルスキャン回路に対する短縮スキャンシフト法

短縮スキャンシフト法をパーシャルスキャン回路に応用した方法[70],[71]について述べる．この方法は，順序回路が与えられたときに，スキャンチェーンを構成し，テスト系列を生成する．目的は，できるだけ短いテスト系列を得ることと，できるだけ高い故障検出率を得ることである．パーシャルスキャン回路[72]-[77]はフルスキャン回路と異なり，制御や観測のできないFFがあるため，元のテストベクトルと同じ故障検出率が常に得られるとは限らない．提案方法は，テスト系列を短くするための短縮スキャンシフト法と，故障検出率を高くするためのいくつかの方法を併せて用いる．本論文では，スキャンFFの数は予め与えられていると仮定する．

５．１　パーシャルスキャン回路の設計

５．１．１　スキャンフリップフロップの選択と配列

スキャンFFの選択とスキャンチェーン上の配列を決めるときには，テスト系列ができるだけ短くなるように，即ち，短縮スキャンシフト法がより効果的になるようにする．テストベクトルに対し，制御や観測の必要なFFが求められた後，制御と観測の必要性の高いFFをスキャンFFに選び，それらをスキャンチェーン上に配列する．スキャンチェーン上の配列は，フルスキャン回路の場合と同様に，制御の必要性の高いFFをスキャン入力側に，観測の必要性の高いFFをスキャン出力側になるように配列する．FFの制御や観測の必要性を表すVC，VOから計算されるW_p，W_mを各FF毎に定義する．

$$W_p = VC(FF_i) + VO(FF_i) \qquad (5.1)$$

$$W_m = VC(FF_i) - VO(FF_i) \qquad (5.2)$$

W_pが大きいことは，そのFFが制御と観測の必要性の大きいことを表し，W_mが大きいことは，そのFFが制御の必要性が観測の必要性よりも高いことを表す．パーシャルスキャン回路を設計する際には，W_pの大きい順にスキャンFFに選び，W_mの大きい順にスキャン入力側からFFを配列し，スキャンチェーンを構成する．

例5.1：回路内の5つのFFの中から，3つのスキャンFFを選びパーシャルスキャン回路を設計することを考える．与えられたテストベクトルに対する制御と観測の必要性を調べた結果が表5.1(a)のようであったとする．" 設定すべき値 "の欄では，テストベクトル中のFFの値を示してあり，XのFFは制御の必要がないことを表す．" 観測すべきFF "の欄はテストベクトルを印加した後の故障の影響が伝搬した結果，観測の必要があるFFを示している．表5.1(a)より求められた，VC，VO，W_p，W_mを表5.1(b)に示す．この結果，FF_a, FF_c, FF_dをスキャンFFに選び，スキャン入力側から，FF_c, FF_d, FF_aの順に配列する．

表5.1　(a)テストベクトルとFFの制御・観測の必要性
(b)各FFに対するVC, VO, Wp, Wm

(a)

テスト ベクトル	設定すべき値 (FFa, FFb, FFc, FFd, FFe)	観測すべきFF (FFa, FFb, FFc, FFd, FFe)
t1	(0, X, 0, 1, X)	(⊕, -, -, -, ⊕)
t2	(X, 1, 0, 0, X)	(⊕, -, -, ⊕, -)
t3	(1, X, 1, X, 0)	(-, ⊕, ⊕, -, -)
t4	(1, 1, 0, 1, X)	(-, ⊕, -, -, ⊕)
t5	(0, X, 0, 0, 1)	(⊕, -, -, ⊕, -)

X：ドントケア　⊕：故障の影響の伝搬有

(b)

	VC	VO	Wp	Wm
FFa	4	3	7	1
FFb	2	2	4	0
FFc	5	1	6	4
FFd	4	2	6	2
FFe	2	2	4	0

5．1．2　パーシャルスキャン回路の実装

設計するパーシャルスキャン回路は，スキャンシフト操作時に，非スキャンFFの状態が保持されるような特徴を持つ．これは，テストベクトルの順序を決定するときに，非スキャンFFの状態とテストベクトルの値を比較するためで，スキャンシフト操作時の非スキャンFFの状態遷移の計算を避けることができる．前述の特徴を持つ回路として，様々な実装が可能であるが，ここでは図5.1のような回路を提案する．この回路は，付加入力線として，スキャン入力SIとモード制御MCの2つで済み，回路チップの外部入力ピン数に厳しい制約がある場合に特に有効である．FF_1，FF_2，FF_3がスキャンFFで，FF_4，FF_5が非スキャンFFである．スキャン回路の動作としては，モード制御MCの値が1のとき，すべてのFFにクロックが入力し，このとき，各FFには組合せ部分からの信号が入力するような，通常の順序回路として動作する．モード制御MCの値が0のときは，スキャン入力SIからの値がスキャンFFを順にシフトしてスキャン出力SOから出力されるスキャンシフトの動作で，非スキャンFFは，クロックの値がすべて0となることによってその状態を保持する．

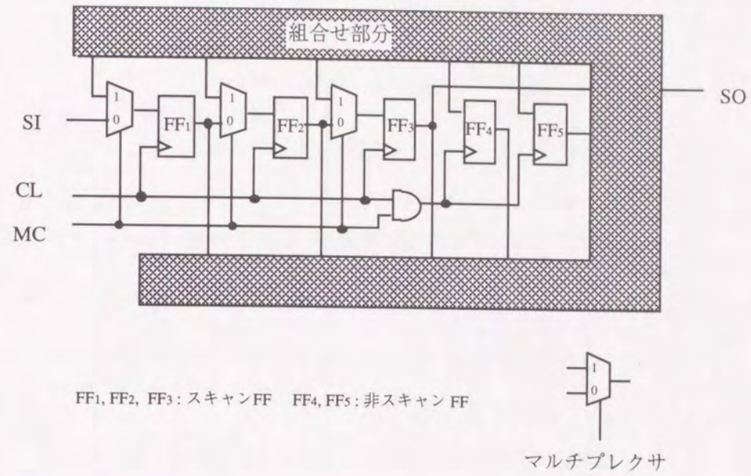


図5.1　パーシャルスキャン回路

5．2　テスト系列生成

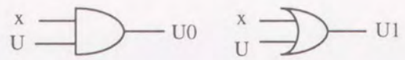
5．2．1　概要

パーシャルスキャン回路におけるテスト系列生成は，1) 初期化系列生成，2) 組合せ部分のテストベクトルとスキャンシフト操作時の入力系列からなるテスト系列の生成，3) 順序回路に対するテスト系列生成アルゴリズムによるテスト系列生成の3段階に分けて行う．これらの操作で得られた系列を連結し，1つのテスト系列とする．いずれのテスト系列生成においても，故障検出率ができるだけ高くなるような方法を用いる．テスト系列長に関しては，制御や観測の必要なFFを選択してスキャンシフトを行うことにより，短いテスト系列長を実現する．この節では，制御や観測の必要なFFを調べ，パーシャルスキャン回路を設計した後のテスト系列生成について述べる．

パーシャルスキャン回路において，テストベクトルが達成するものと同じ故障検出率が得られないのは，非スキャンFFが存在し，これを直接制御・観測できないためである．非スキャンFFの値が特定の0または1でなければ顕在化されない故障や，顕在化された故障の影響が非スキャンFFにしか伝搬しない故障は，スキャンシフト操作を用いても検出することが困難である．以下で述べる方法は，このような故障をできるだけ検出することを目指している．

5．2．2　初期化系列

順序回路をテストする場合，テストを始める前の回路の状態(初期状態)は未知であるので，これを適



x→0 ⇒ U0→0
x→1 ⇒ U0→U
x→0 ⇒ U1→U
x→1 ⇒ U1→1

図5.2　2入力ゲートでのUを含む演算

表5.2　(a)AND演算　(b)OR演算　(c)NOT演算

(a)

	0	1	x	U	U0	U1
0	0	0	0	0	0	0
1	0	1	x	U	U0	U1
x	0	x	x	U0	U0	x
U	0	U	U0	U	U0	U
U0	0	U0	U0	U0	U0	U0
U1	0	U1	x	U	U0	U1

(b)

	0	1	x	U	U0	U1
0	0	1	x	U	U0	U1
1	1	1	1	1	1	1
x	x	1	x	U1	x	U1
U	U	1	U1	U	U	U1
U0	U0	1	x	U	U0	U1
U1	U1	1	U1	U1	U1	U1

(c)

0	1	x	U	U0	U1
1	0	x	U	U1	U0

当なある状態に遷移させるための入力系列(初期化系列)が必要である．フルスキャン回路では，スキャンシフト操作によってすべてのFFが制御できるため，初期化系列は必要でないが，パーシャルスキャン回路では，非スキャンFFを初期化するための初期化系列を生成しなければならない．ここでは文献[78]に基づき，6つの信号値を用いた方法を導入する．その信号値とは，0，1，x，U，U0，U1である．信号値0，1は低電位，高電位に相当する通常の論理値，xはテスト生成における値の割当がまだ決まっていない値を表す．信号値Uは初期のFFの値で0，1どちらとも分からない未知の論理値を表す．信号値U0，U1はxとUの論理演算を行ったときに生じる値で，将来xの信号値に0または1を割り当てたときに，U0はUまたは0に，U1はUまたは1になることを示している．例えば，図5.2のような2入力ANDゲートの一方の入力がxで，もう一方がUのとき，その出力はU0となる．これは，テスト生成においてxに0，1のどちらを割り当てても，出力は1にはならないことを意味する．xに0を割り当てたときには，U0は0に変わり，xに1を割り当てたときには，U0はUに変わる．6つの信号値のAND，OR，NOT論理演算を表5.2に示す．

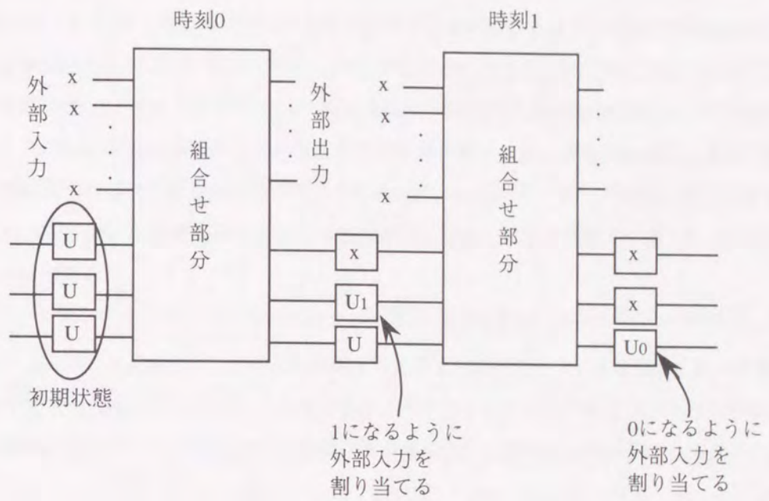


図5.3　時間展開回路を用いた初期化

初期化系列生成では，図5.3に示されるような時間展開した回路を用いる．まず始めに，FFにU，外部入力にxを割り当て，論理シミュレーションを行う．もし，あるFFの入力線の値(次状態のFFの値)がU0なら，それが0になるように外部入力の値を割り当て，また，U1なら1になるように外部入力の値を割り当てる．入力線の値がxであるFFは，外部入力を適当に割り当てて，0または1に初期化する．もし，入力線の値がUであるFFがあれば，そのFFはその時刻での初期化が不可能であるので，図5.3に示されるように，時刻を進めて同様の操作を行い初期化する．

5．2．3　テストベクトルの順序

パーシャルスキャン回路においては，テストベクトルを印加する順序は，テスト系列長に影響を与えるだけでなく，達成される故障検出率にも影響を与える．即ち，スキャンシフト操作による制御や観測の不可能な非スキャンFFの状態によって，検出される故障が異なる．そこで，テストベクトルの順序を決めるときには，テストベクトルを印加した後の回路の状態と，次に印加するテストベクトルの値を比較し，故障検出率ができるだけ高くなるようにする．もし，すべての非スキャンFFの値がテストベクトル中のFFの値と一致するなら，そのテストベクトルで顕在化される故障が，得られたテスト系列によっても必ず顕在化されることになる．

テストベクトルを印加した後の非スキャンFFの値と，テストベクトルが要求するFFの値の一致を調べるために $nz(t \oplus S)$ を定義する．$nz(t \oplus S)$ は非スキャンFFに関して，回路の状態 S とテストベクトル t の値

の排他的論理和を計算したときの0の個数とする．例えば，状態 $S(FF_1, FF_2, FF_3)=(1,1,0)$，テストベクトル $t(FF_1, FF_2, FF_3)=(0,1,0)$のとき，$FF_2$，$FF_3$の値が一致するので，$nz(t \oplus S)$ は2となる．

テストベクトルの順序を決める際には，まだ選ばれていないテストベクトルtと現状態Sについて$nz(t \oplus S)$を計算し，$nz(t \oplus S)$が最も大きいテストベクトルを選択する．提案方法は，テストベクトルの順序を1番目から順に決めていき，途中でバックトラックを起こさない方法である．各テストベクトルの間に必要なスキャンシフトの長さは，スキャン制御数とスキャン観測数に従って一意に決まる．

5．2．4　順序回路のテスト生成の応用

初期化系列と短縮スキャンシフト法によるテスト系列によって，未検出故障が残っている場合には，順序回路のテスト生成器を用いてさらにテスト系列を生成し，先のテスト系列に付加する．順序回路のテスト生成器を用いることの利点は，複数時刻の外部入力の制御によって，FFへの値の設定や故障

Procedure: Test sequence generation for partial scan circuits
Generate test vectors for the combinational part ;
Find essential faults for each test vector ;
Analyze which FFs are required to be controlled or observed ;
Calculate VC, VO, W_p and W_m ;
Select scanned FFs ;
Arrange the FFs in a scan chain ;
Calculate NSC and NSO for the essential faults ;
L_r = Generate_initilization_sequence ( ) ;
for (j = 1 to v) /* v is the number of test vectors */
{
　　Select test vector t_j according to the selection rule ;
　　$L_r = L_r$ + { Scan shift operations required before t_j} + t_j ;
}
$L_r = L_r$ + { Scan shift operations required after t_v };
if (L_r achieves 100% fault efficiency)
　　exit;
Generate test sequence L_s to detect faults undetected by L_r using a sequential circuit test generator ;
Add L_s to L_r ;

図5.4　テスト系列生成アルゴリズム

の影響の伝搬を行うことであり，非スキャンFFに対しても値の制御が可能となることである．従って，スキャンシフト操作をどのように工夫しても，検出することができない故障を，容易に検出できることもある．反面欠点としては，大きな計算時間が必要となることや，生成されるテスト系列長が長くなること，などの問題がある．短縮スキャンシフト法を用いる前に順序回路のテスト生成器を用いてテスト系列を生成することも考えられるが，その場合には，短縮スキャンシフト法で容易に検出できる故障までも，順序回路のテスト生成器によって検出されることとなり，計算時間，テスト系列長の両面で非効率的である．

組合せ部分のテストベクトル生成から始まるテスト系列生成の全体のアルゴリズムを図5.4に表す．

5．3　実験結果

パーシャルスキャン回路に対して短縮スキャンシフト法を応用した提案方法をプログラム化し，Sun-SS/Classicワークステーション上で，ISCAS '89ベンチマーク回路[52]に対して実験を行った．表5.3には，短縮スキャンシフト法を用いた結果("R"の行)，フルスキャンシフト法を用いた結果("F"の行)，短縮スキャンシフト法の結果のフルスキャンシフト法の結果に対する割合("R/F"の行)を示す．"平均"の行は，各回路の"ratio R/F"の平均を示す．"回路"，"FF"，"S-FF"，"V"，"R"の各列はそれぞれ，回路名，FFの総数，スキャンFFの数，組合せ部分のテストベクトル数，冗長故障数を表す．スキャンFF数は，総FF数の3分の2とした．テストベクトル生成には，文献[69]のテスト圧縮技法を採り入れ，冗長故障はそのときに見つけられたものである．"系列長"の列はテスト系列長を表し，"短縮"は，組合せ部分のテストベクトルを基に，短縮スキャンシフト法またはフルスキャンシフト法を用いた結果である．"順序"は，文献[79]の方法に基づく順序回路のテスト生成器を用いて生成したテスト系列長を示す．"合計"は"短縮"と"順序"の和で，最終的に得られたテスト系列長である．"検出率"は故障検出率であり，冗長故障を除いた全故障数に対する検出故障数の割合を示し，"短縮"と"合計"はテスト系列長の"短縮"と"合計"にそれぞれ対応した結果である．"時間"は計算時間を示し，"順序"は順序回路のテスト生成器を用いてテスト系列を生成した時間，"合計"は"順序"を含むすべての計算時間である．表5.3より短縮スキャンシフト法を用いることによって，フルスキャンシフト法を用いるより短いテスト系列を得られることが分かる．ただし，いくつかの回路では，フルスキャンシフト法の方が高い故障検出率を得られる．順序回路のテスト生成器によるテスト系列長は，未検出故障数とともに増大するため，"短縮"の故障検出率が低いと，"順序"のテスト系列長は長くなる．各項目の"合計"を比較すると，短縮スキャンシフト法を用いた場合には，フルスキャンシフト法を用いた場合より短いテスト系列(平均0.75倍)でほぼ同じ故障検出率(平均1.01倍)を得ることができると分かる．計算時間のほとんどは順序回路のテスト生成法による計算時間であるので，組合せ部分のテストベクトルより得られたテスト系列によって高い故障検出率を得ることが，順序回路のテスト生成の割

表5.3　短縮スキャンシフト法とフルスキャンシフト法による実験結果

回路		FF	S-FF	V	R	系列長 短縮	系列長 順序	系列長 合計	検出率(%) 短縮	検出率(%) 合計	時間(s) 順序	時間(s) 合計
s208	R	8	6	28	0	161	51	212	84.7	96.5	32.1	34.2
	F					203	36	239	90.8	96.9	30.0	32.0
	R/F					0.79	1.42	0.89	0.93	1.00	1.07	1.07
s298	R	14	10	26	0	207	60	267	95.4	100.0	3.7	7.3
	F					297	76	373	93.3	97.9	53.5	57.6
	R/F					0.70	0.79	0.72	1.02	1.02	0.07	0.13
s344	R	15	10	18	0	140	46	186	89.4	100.0	15.9	19.3
	F					210	41	251	91.7	100.0	3.0	6.6
	R/F					0.67	1.12	0.74	0.97	1.00	5.30	2.94
s382	R	21	14	27	0	148	194	342	75.5	94.0	168.6	174.0
	F					420	105	525	81.4	96.1	54.0	62.3
	R/F					0.35	1.85	0.65	0.93	0.98	3.12	2.79
s386	R	6	4	65	0	309	62	371	93.3	97.4	58.9	65.1
	F					332	113	445	85.2	96.6	94.2	100.7
	R/F					0.93	0.55	0.83	1.10	1.01	0.63	0.65
s420	R	16	11	45	0	432	170	602	76.2	84.6	542.2	554.0
	F					552	0	552	83.3	83.3	375.9	385.7
	R/F					0.78		1.09	0.91	1.02	1.44	1.44
s444	R	21	14	27	14	260	73	333	90.4	97.2	114.1	123.3
	F					420	0	420	96.0	96.0	42.1	50.7
	R/F					0.62		0.79	0.94	1.01	2.71	2.43
s641	R	19	13	27	0	298	15	313	99.2	100.0	2.7	11.3
	F					392	72	464	96.0	100.0	14.2	22.2
	R/F					0.76	0.21	0.67	1.03	1.00	0.19	0.51
s820	R	5	4	101	0	490	133	623	89.2	98.9	266.1	295.8
	F					510	175	685	86.7	99.3	245.7	275.5
	R/F					0.96	0.76	0.91	1.03	1.00	1.08	1.07
s838	R	32	22	78	0	1299	559	1858	71.7	81.6	570.9	656.5
	F					1817	0	1817	81.3	81.3	708.3	775.2
	R/F					0.71		1.02	0.88	1.00	0.81	0.85
s953	R	29	20	79	0	581	0	581	99.9	99.9	3.4	49.0
	F					1669	0	1669	100.0	100.0	0.0	56.2
	R/F					0.35		0.35	1.00	1.00		0.87
s1238	R	18	12	136	69	395	52	447	97.8	99.8	28.6	89.8
	F					1781	23	1804	93.9	94.8	485.2	563.9
	R/F					0.22	2.26	0.25	1.04	1.05	0.06	0.16
s1423	R	74	50	34	14	1257	280	1537	89.3	98.4	855.1	1072.5
	F					1787	280	2067	83.4	95.9	2278.8	2476.4
	R/F					0.70	1.00	0.74	1.07	1.03	0.38	0.43
s1488	R	6	4	108	0	509	71	580	96.4	98.6	791.1	841.5
	F					545	191	736	91.5	98.3	1125.8	1178.7
	R/F					0.93	0.37	0.79	1.05	1.00	0.70	0.71
s5378	R	179	120	108	40	10555	593	11148	80.9	93.5	39596.5	48739.9
	F					13197	0	13197	92.2	92.2	1829.2	4786.5
	R/F					0.80		0.84	0.88	1.01	21.65	10.18
平均						0.69	1.03	0.75	0.99	1.01	2.80	1.75

表5.4　テストベクトルの順序とFFの配列の効果

回路		S-FF	系列長 短縮	系列長 順序	系列長 合計	割合 短縮	割合 合計	検出率(%) 短縮	検出率(%) 合計
s208	U	6	169	57	226	1.00	1.00	84.7	96.5
	M	6	169	56	225	1.00	1.00	89.1	97.0
	L	6	161	51	212	0.95	0.94	84.7	96.5
s298	U	10	224	59	283	1.00	1.00	92.9	97.8
	M	10	224	20	244	1.00	0.86	98.2	99.7
	L	10	207	60	267	0.92	0.94	95.4	100.0
s344	U	10	146	53	199	1.00	1.00	84.2	99.8
	M	10	146	45	191	1.00	0.96	93.1	100.0
	L	10	140	46	186	0.96	0.93	89.4	100.0
s382	U	14	379	63	442	1.00	1.00	84.8	96.3
	M	14	379	73	452	1.00	1.02	86.7	96.1
	L	14	148	194	342	0.39	0.39	75.5	94.0
s386	U	4	310	109	419	1.00	1.00	83.2	96.2
	M	4	310	83	393	1.00	0.94	86.0	97.4
	L	4	309	62	371	1.00	1.00	93.3	97.4
s420	U	11	451	17	468	1.00	1.00	80.0	80.4
	M	11	451	19	470	1.00	1.00	77.8	78.2
	L	11	432	170	602	0.96	0.96	76.2	84.6
s444	U	14	297	110	407	1.00	1.00	88.8	96.0
	M	14	297	169	466	1.00	1.14	87.6	95.8
	L	14	260	73	333	0.88	0.88	90.4	97.2
s641	U	13	310	99	409	1.00	1.00	94.4	100.0
	M	13	310	95	405	1.00	0.99	93.3	99.8
	L	13	298	15	313	0.96	0.96	99.2	100.0
s820	U	4	479	215	694	1.00	1.00	85.6	99.3
	M	4	479	65	544	1.00	0.78	96.6	100.0
	L	4	490	133	623	1.02	1.02	89.2	98.9
s838	U	22	1349	91	1440	1.00	1.00	71.8	73.7
	M	22	1349	131	1480	1.00	1.03	71.9	74.9
	L	22	1299	559	1858	0.96	0.96	71.7	81.6
s953	U	20	510	54	564	1.00	1.00	98.3	100.0
	M	20	510	47	557	1.00	0.99	98.4	100.0
	L	20	581	0	581	1.14	1.14	99.9	99.9
s1238	U	12	493	64	557	1.00	1.00	92.1	94.8
	M	12	493	57	550	1.00	0.99	92.2	94.8
	L	12	395	52	447	0.80	0.80	97.8	99.8
s1423	U	50	1135	309	1444	1.00	1.00	84.7	96.2
	M	50	1135	301	1436	1.00	0.99	84.3	97.3
	L	50	1257	280	1537	1.11	1.11	89.3	98.4
s1488	U	4	508	183	691	1.00	1.00	90.2	98.1
	M	4	508	88	596	1.00	0.86	94.8	98.1
	L	4	509	71	580	1.00	1.00	96.4	98.6
s5378	U	120	11840	0	11840	1.00	1.00	91.9	91.9
	M	120	11840	0	11840	1.00	1.00	92.7	92.7
	L	120	10555	593	11148	0.89	0.89	80.9	93.5
平均	U					1.00	1.00	87.2	94.5
	M					1.00	0.97	89.5	94.8
	L					0.93	0.93	88.6	96.0

表5.5　他の方法との比較

		提案法			文献[62]			文献 [63]		
回路	FF	S-FF	系列長	検出率	S-FF	系列長	検出率	S-FF	系列長	検出率
s208	8	6	212	96.5	7	177	90.8	3	336	90.2
s298	14	10	267	100.0	1	211	97.1	1	362	94.8
s344	15	10	186	100.0	NA	NA	NA	5	129	98.5
s382	21	14	342	94.0	7	717	98.8	6	697	97.2
s386	6	4	371	97.4	NA	NA	NA	2	369	94.8
s420	16	13	602	84.6	15	939	95.6	7	1057	93.5
s444	21	14	333	97.2	6	592	96.6	6	1197	94.3
s641	19	13	313	100.0	NA	NA	NA	2	211	94.0
s820	5	4	623	98.9	2	728	98.6	2	1019	100.0
s838	32	30	1858	81.6	31	4421	97.9	16	3455	95.1
s953	29	20	581	99.9	3	549	99.5	3	1268	99.9
s1423	74	50	1567	98.4	41	5006	96.8	17	1338	86.5
s5378	179	120	11148	93.5	NA	NA	NA	32	7207	93.4
平均		23.7	1416	95.5	12.6	1482	96.9	7.8	1434	94.8

合を少なくし，全体の計算時間を少なくしたといえる．

スキャンチェーン上のFFの配列と，テストベクトルの順序付けの効果を見るための実験の結果を表5.4に示す．各回路の" U "の行は，FFの配列もテストベクトルの順序も考えなかった場合，" M "の行は，FFの配列を行わずテストベクトルの順序だけを考えた場合，" L "の行は，FFの配列とテストベクトルの順序の両方を考えた場合の結果を表す．" 回路 "，" S-FF "，" 系列長 "，" 検出率 "，" 時間 "の各列は表5.3と同じ意味の結果を示す．" 割合 "の列は，" 系列長 "の" U "の結果に対する割合である．" U "と" M "を比較することによって，テストベクトルの順序が，得られる故障検出率の向上に役立っていることが分かる．また" M "と" L "を比較することによって，FFの配列がテスト系列を短くすることに役立っていることが分かる．FFの配列とテストベクトルの順序の両方を考えたとき，短いテスト系列で高い故障検出率が得られることが分かる．

他者の研究の結果を表5.5に示す．" 提案法 "が提案方法による結果，" 文献[62] "が文献[62]で報告された結果，" 文献[63] "が文献[63]で報告された結果である．ほとんどの回路について，提案方法によって，テスト系列長が短く，故障検出率も同程度かまたは高くなることがわかる．スキャンFF数では，他者の研究の方が少なくなっているが，提案方法のスキャンFF数は予め与えられた数であり，他者と同様の方法を導入することによって少なくすることが可能であると考えられる．

表5.6　スキャンFF数を変化させた実験の結果

		系列長				検出率(%)		時間(s)	
回路	S-FF	フル	短縮	順序	合計	短縮	合計	順序	合計
s386	6	464	456	0	456	100.0	100.0	0	6.8
s386	5	398	381	18	399	98.2	99.5	10.5	17.0
s386	4	332	309	62	371	93.3	97.4	59.2	65.5
s386	3	266	240	70	310	89.3	95.6	107.4	113.5
s386	2	200	175	103	278	81.7	92.7	144.7	150.4
s386	1	134	114	90	204	75.3	88.5	248.2	253.4
s386	0	68	68	60	128	61.9	76.0	493.5	498.2

" s386 "の回路に対して，スキャンFF数を変化させたときの結果を表5.6に示す．" 系列長 "の欄の" フル "はフルスキャンシフト法により得られたテスト系列長，" 短縮 "は短縮スキャンシフト法により得られたテスト系列長を示す．" 順序 "は，短縮スキャンシフト法で未検出となった故障に対して，順序回路用テスト生成器を用いたときのテスト系列長で，" 合計 "は，" 短縮 "と" 順序 "の和を示す．" 検出率 "の欄の" 短縮 "は短縮スキャンシフト法により得られた故障検出率，" 合計 "は，短縮スキャンシフト法と順序回路用テスト生成器を用いて得られた故障検出率を示す．" 時間 "の欄の" 順序 "は，順序回路用テスト生成器を用いてテスト系列を生成した計算時間，" 合計 "は，短縮スキャンシフト法と順序回路用テスト生成器を用いてテスト系列を生成した計算時間を示す．

この結果より，スキャンFF数が少なくなると，短縮スキャンシフト法による故障検出率が低くなり，それに伴って順序回路のテスト生成の計算時間が長くなる．

５．４　まとめ

パーシャルスキャン回路に対して短縮スキャンシフト法を応用したときの，スキャン回路の設計法とテスト系列生成法を述べた．スキャンFFの選択とスキャンチェーン上の配列においては，FFの制御や観測の必要性を指標とした．制御と観測の必要性の和が大きいFFをスキャンFFに選び，制御の必要性が大

きいFFをスキャン入力に近くなるように配列した．テスト系列生成の際には，回路の状態とテストベクトルを比較しテストベクトルの順序を決定することによって，故障検出率が高くなるようにした．パーシャルスキャン回路では，組合せ部分のテストベクトルで検出された故障が，必ずしもテスト系列で検出されないことがあるが，提案方法では，順序回路のテスト生成器を用いることによって，そのような故障も検出でき，最終的には高い故障検出率を達成することができる．

得られる故障検出率は，スキャンFF数に依存するが，提案方法では，スキャンFF数は予め与えられると仮定したため，常に最適のFFが選択された訳ではない．ある回路に対しては，もっと少ないスキャンFF数で高い故障検出率が得られたかもしれない．またある回路に対しては，もっと多くのスキャンFFが必要であったかもしれない．今後の課題としては，ある目標の故障検出率が与えられたときに，最適なスキャンFFを選択し，短いテスト系列を生成する方法を開発する必要がある．

第6章　リタイミングを応用した短縮スキャンシフト法

リタイミングとは，FFの位置を変えることによって出力関数を変えずに回路のタイミング動作を変える技術で，最近様々な目的で使用されている[24]-[28]．ここでは，短縮スキャンシフト法にリタイミングを応用した方法[80]について述べる．対象回路はフルスキャン回路を仮定しており，目的は，短いテスト系列を得ることである．ただし，フルスキャン回路を仮定していることは，与えられたテストベクトルが達成する故障検出率と同じ故障検出率が，得られたテスト系列によって達成できることを意味している．フルスキャンシフト法を用いたときには，リタイミングによりFF数を減らすことがテスト系列の短縮につながり，そのことは容易に理解できる．一方短縮スキャンシフト法を用いたときには，FF数を減らすだけでなく，FFの制御や観測の必要性が小さくなるようなリタイミングを行わなければならない．ここでは，リタイミングを，FFの減少する場合と，FF数が変わらない場合に分けて説明する．また，リタイミングを行った結果，FFの制御や観測の必要性が小さくなるような回路構造を，組合せ部分のテストベクトルを基に見つける方法を述べる．リタイミングによって回路構造が変化したとき，一般的には，前の回路で有効であったテストベクトルが無効になるが，ここでは，元のテストベクトルを変更し，リタイミングによって得られた回路においても，元の回路に対するものと同じ故障検出率を得る方法を述べる．

6．1　概要

リタイミングは，ゲートレベルで記述された回路において，出力の論理関数を変えることなくFFを移動する技術である．リタイミングは，回路内のパスの遅延を減らしたり，FF数を減らすために利用することができる．例えば，図6.1 (a) の回路において，ゲートG_1の入力にあるFFを，図6.1 (b) のようにゲー

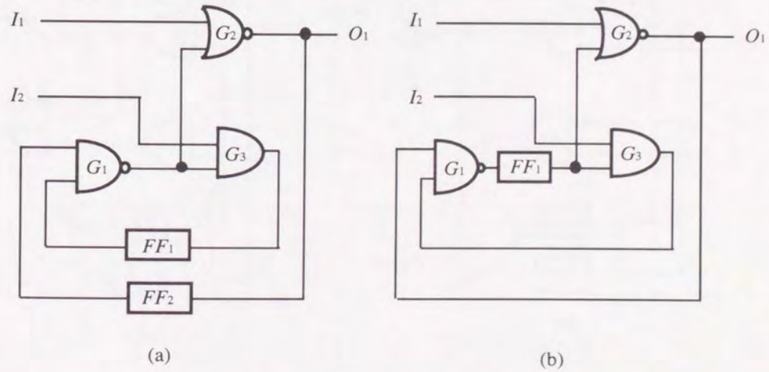


図6.1　(a)リタイミング前の回路と　(b)リタイミング後の回路

トの出力に移動して1つにしても，その回路の出力関数は変わらない．

リタイミングによってFF数が減ることは，スキャン回路のテストにおいては，スキャンシフト数が減ることを意味し，結果としてテスト系列長を短くすることとなる．従って，フルスキャンシフト法によるテストでは，FF数を最小にすることがテスト系列を最短にすることにつながる．ところが短縮スキャンシフト法では，テスト系列が短くなるのはFF数が減る場合だけでなく，たとえFF数が変わらなくてもテスト系列を短くできる場合がある．図 6.2 (a) と図 6.2 (b) は同じ入出力関数を実現する回路で，FF数も変わらないが，故障を検出するために必要となるFFの制御の必要性が異なる．図 6.2 (a) においてゲートG_4の出力線の0縮退故障を検出するためには，FF_1を1に設定するだけでよいが，図 6.2 (b) ではFF_1とFF_2の2つのFFを1に設定しなければならない．このようにリタイミングにより，FFの制御や観測の必要性に違いが生じるときには，短縮スキャンシフト法におけるスキャンシフト数に違いが生じる．即ち，短縮スキャンシフト法では，FF数を少なくするだけでなく，FFの制御や観測の必要性が小さくなるように，リタイミングを行う必要がある．

次節以下ではリタイミングについて，FF数が減る場合と，FF数が変わらない場合に分けて議論する．FF数が減る場合には，生成したテストベクトルに拘わらず，テスト系列は必ず短くなる．一方，FF数が変わらない場合には，リタイミングによってFFの制御や観測の必要性が元の回路より少なくなった場合にのみ，テスト系列が短くなる．つまり，FF数が変わらないリタイミングでは，テスト系列が短くなることは，テストベクトルのFFの値に依存している．そこで，提案方法では，まず最初に，FF数ができるだけ少なくなるようなリタイミングを行う．次に，得られた回路の組合せ部分に対して，テストベクトルを生成する．生成したテストベクトルに対して，短縮スキャンシフトの技法を用いて，各FFの制御や観測の必要性を調べる．その結果から，テスト系列が短くなるように，FF数が変わらないリタイミング

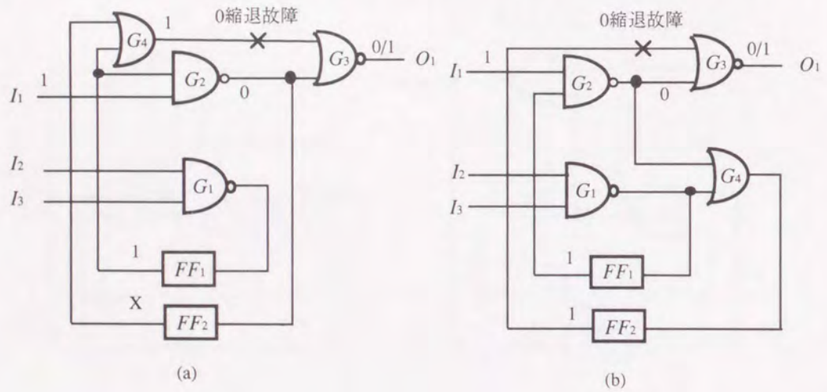


図6.2　リタイミングによるFFの制御の違い

を行う．このリタイミングによってテスト系列が短くなるかどうかは，前に生成されたテストベクトルに依存して決まる．ここで一つ問題となることは，リタイミングを行った後の回路では，元のテストベクトルが必ずしも有効でないことである．つまり，リタイミングによって回路構造が変化するため，元のテストベクトルをそのまま用いたときには，未検出故障が現れる場合がある．提案方法では，故障検出率を低下させることなく，元のテストベクトルを変更する方法を考える．これによって，新たにテストベクトルを生成するより計算時間を節約する．

6．2　フリップフロップ数の減少するリタイミング

FF数が減少するようなリタイミングについて述べる．

タイプI－ゲートの入出力におけるリタイミング：

図6.3(a)のように，ゲートのすべての入力線がFFと直接つながっているとき，FFをゲートの出力に移動し，一つとすることができる．図6.3(a)から図6.3 (b) への変換をタイプIリタイミングという．逆に図6.3 (b) から図6.3 (a) への変換も可能であるが，FF数が増加するため，このようなリタイミングは考慮に入れない．1入力ゲートに対してもタイプIリタイミングは可能であり，従って，図6.4(a)のようにANDゲートとFFの間にNOTゲートが存在しても，図6.4(b)のように変換することができる．図6.5(a)のようにゲートの一部の入力線にしかFFが接続していない場合，図6.5(b)のように，ゲートを付加した後

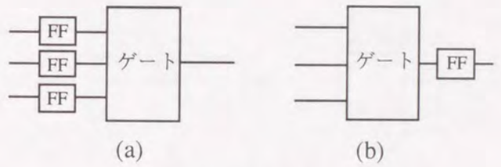


図6.3　タイプIリタイミング

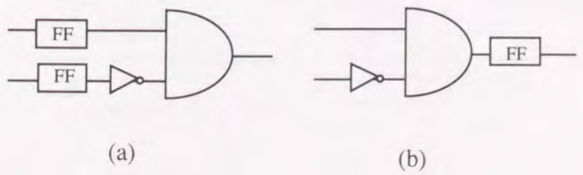


図6.4　NOTゲートを含むタイプIリタイミング

にタイプIリタイミングを行うことが可能である．付加するゲートの種類は，元のゲートの種類に応じて一意に決定でき，元のゲートがAND，NANDならANDゲートを，元のゲートがOR，NORならORゲートを付加すればよい．

タイプII－分岐の幹と枝におけるリタイミング：

図6.6のように，分岐の幹にあるFFを，すべての分岐の枝に移動することが可能である．図6.6(a)から図6.6(b)への変換に相当する，分岐の幹から分岐の枝へFFを移動させる変換をタイプIIリタイミングという．逆に，分岐の枝から分岐の幹へFFを移動させる変換(図6.6(b)から図6.6(a))では，FF数が増加するため，そのようなリタイミングは考えない．一部の分岐の枝のみにFFがつながっている場合，その部分に

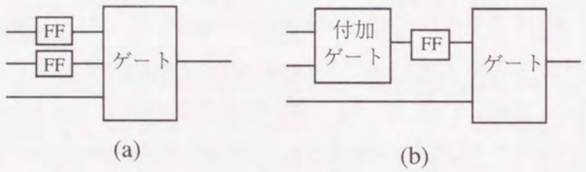


図6.5　ゲートを付加したタイプIリタイミング

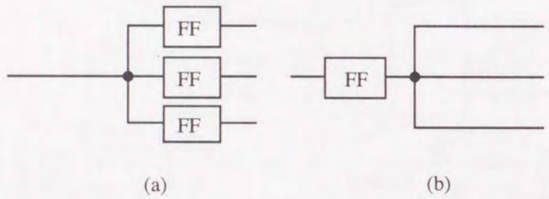


図6.6　タイプIIリタイミング

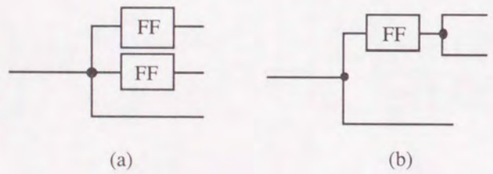


図6.7　タイプIIリタイミング

のみタイプIIリタイミングを行うことが可能である(図6.7参照)．

6．3　フリップフロップ数の変わらないリタイミング

[定義6.1] α構造：図6.8(a)のように，1つのゲートと2つのFFからなり，1つのFFとゲートが同じ幹を持つ別の分岐の枝上にあり，もう1つのFFがそのゲートの出力につながっている回路構造をα構造という．説明上，分岐の枝からつながるFFを$FF_{\alpha 1}$，ゲートからつながるFFを$FF_{\alpha 2}$とする．

[定義6.2] β構造：図6.8(b)のように，1つのゲートと2つのFFからなり，1つのFFの出力が2本に分岐し，その一方にゲートがつながり，もう1つのFFがそのゲートの入力につながっている回路構造をβ構造という．説明上，分岐のあるFFを$FF_{\beta 1}$，分岐のないFFを$FF_{\beta 2}$とする．

タイプIII－α構造とβ構造におけるリタイミング：

α構造をβ構造に変換するリタイミングをタイプIII後方リタイミング，β構造をα構造に変換するリタイミングをタイプIII前方リタイミングという．タイプIIIリタイミングにおいては，FF数は変化しない．α構造やβ構造以外についても，タイプIIリタイミングやゲートを付加した後，タイプIIIリタイミングを行

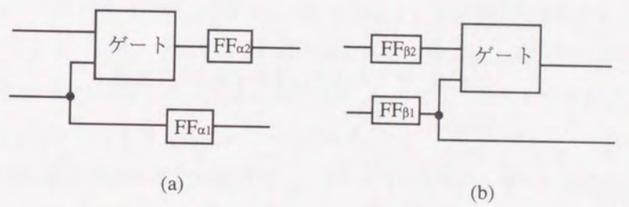


図6.8　(a)α構造と　(b)β構造

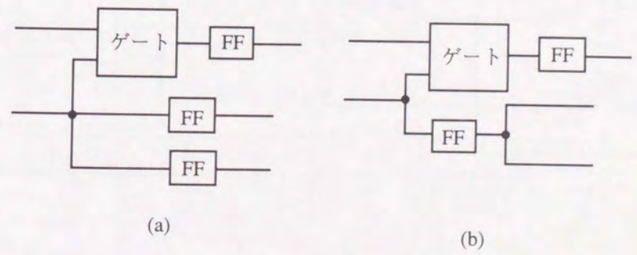


図6.9　タイプIIリタイミングによるα構造への変換

うことができる．例えば，分岐の枝に2つ以上のFFがつながっているときには，タイプIIリタイミングを行うことによってα構造に変換することができる（図6.9参照）．α構造において，ゲートの入力が3本以上のときには，ゲートを付加することによって，α構造への変換が可能である（図6.10参照）．

6．4　スキャンシフト数の減少

ここでは，FF数が変わらないタイプIIIリタイミングを行ったときに，スキャンシフト数が減少するために必要な条件について説明する．

図6.11(a)の回路内のゲートG_2の1縮退故障（故障f）について考える．故障fはテストベクトル$t_\alpha=(I_1, I_2, I_3, FF_{\alpha1}, FF_{\alpha2})=(0, 1, X, 0, 1)$によって検出されるため，スキャンチェーン上のFFの配列を入力側から$FF_{\alpha1}$-$FF_{\alpha2}$の順と仮定すると，$FF_{\alpha1}$と$FF_{\alpha2}$を制御するために長さ2のスキャンシフトが必要である．次にタ

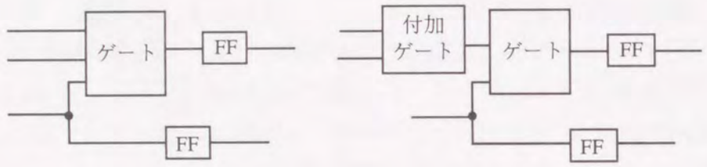


図6.10　ゲート付加によるα構造への変換

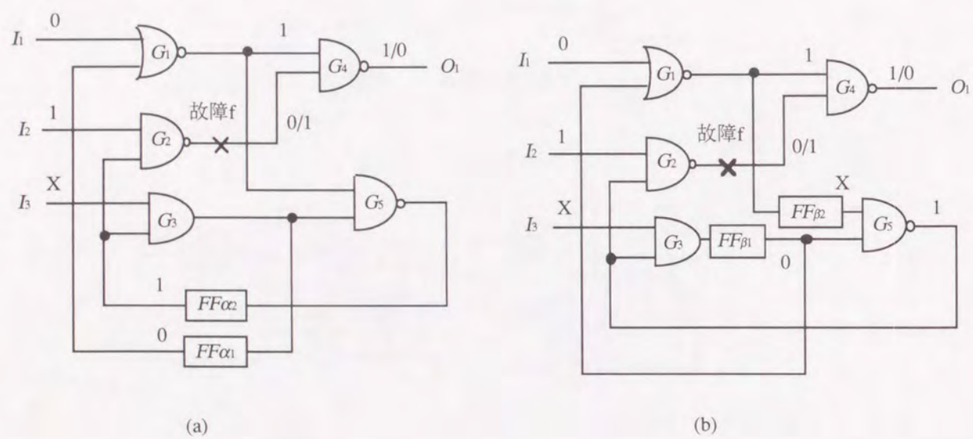


図6.11　リタイミングとFFの制御の違い

イプIII後方リタイミングを行った図6.11(b)の回路において同じ故障fを検出することを考える．故障fを検出するためのテストベクトルは$t_\beta=(I_1, I_2, I_3, FF_{\beta1}, FF_{\beta2})=(0, 1, X, 0, X)$となり，$FF_{\beta1}$を制御するだけでよく，スキャン制御数が1となり，必要なスキャンシフト数が小さくなる．この例では，リタイミング前のテストベクトルのFFの値が(0, 1)であったものが，リタイミング後に(0,X)になり，$FF_{\beta2}$の制御の必要がなくなったことが，スキャン制御数減少の原因である．仮にFFの値が$(FF_{\alpha1}, FF_{\alpha2})=(1, 1)$，(1, 0)の場合を考えると，リタイミング後のFFの値はそれぞれ$(FF_{\beta1}, FF_{\beta2})=(1, 0)$，(1, 1)となり，FFの制御の必要性は変わらず，スキャンシフト数も変化しない．また，リタイミング前のテストベクトル中のFFの値が$(FF_{\alpha1}, FF_{\alpha2})=(0, 0)$のときには，$FF_{\beta1}$と$FF_{\beta2}$をどのように割り当てても，回路内の信号線にリタイミング前と同じ値を割り当てることはできないことが分かる．

タイプIIIリタイミングによって，テスト系列が短くなる場合の本質は，元の回路では0または1に制御しなければならないFFの値が，リタイミング後の回路ではドントケアとなる場合である．表6.1は，タイプIII後方リタイミングを行ったときに，テストベクトル中の$FF_{\beta1}$, $FF_{\beta2}$の値がどのように変わるかについて，ゲートの種類毎にまとめた結果である．表中で2組の値が示してあるのは，リタイミング後に，リタイミング前と同じ値の割り当てができない場合で，いずれか一方を選択するとによって，$FF_{\alpha1}$または$FF_{\alpha2}$の値が保持されることを示している．選択する値によっては，スキャンシフト数が減る場合もある．表6.1より分かることは，ゲートの種類がXOR, XNORのときには，どのような$(FF_{\alpha1}, FF_{\alpha2})$の組合せでも，FFの値がドントケアにならない，つまり，スキャンシフト数が小さくならないことである．

タイプIII前方リタイミングについて考えると，全てのゲートの種類と，$(FF_{\beta1}, FF_{\beta2})$の値の組合せについて，$FF_{\alpha1}$, $FF_{\alpha2}$の値がドントケアとなることはあり得ない．つまり，テスト系列が短くなるのは，タイプIII後方リタイミングを行った場合に限ることが分かる．

各テストベクトル毎に決まるFFの制御の必要性と，リタイミング後のスキャン制御数の減少について考える．リタイミング後にスキャン制御数が減るためには，リタイミング前の$FF_{\alpha1}$と$FF_{\alpha2}$は，どちらも制御の必要のあるFFでなければならない．なぜなら，リタイミング前に制御の必要がないのなら，リタイミング後も制御の必要がなくスキャンシフト数は変わらないためである．また，リタイミング前の$FF_{\alpha2}$の値は，ゲートの出力制御値でなければならない．出力制御値とは，ゲートの制御値によって一意

表6.1　$FF_{\beta1}$と$FF_{\beta2}$の値

$FF_{\alpha1}$, $FF_{\alpha2}$	AND	NAND	OR	NOR	XOR	XNOR
00	0X	11 or 0X	00	01	00	01
01	11 or 0X	0X	01	00	01	00
11	11	10	1X	00 or 1X	10	11
10	10	11	00 or 1X	1X	11	10

に決まるゲートの出力の値のことで，例えば，AND，NORゲートに対しては0，OR，NANDゲートに対しては1である．もし$FF_{\alpha 2}$の値が出力制御値でなければ，リタイミング後は，$FF_{\beta 1}$と$FF_{\beta 2}$の両方の制御が必要でスキャンシフト数は減らない．さらに，$FF_{\alpha 1}$の値がゲートの制御値であるか，または，$FF_{\alpha 1}$が制御の必要のないFFでなければならない．なぜなら，$FF_{\alpha 1}$の値が非制御値で，かつ，$FF_{\alpha 1}$を制御しなければならないとき，リタイミング後は，$FF_{\beta 2}$の値を制御値として制御しなければならないためである．

短縮スキャンシフト法では，必須故障を対象にFFの制御の必要性を調べるので，未検出故障を出さないための次のような条件が求められる．もし，$FF_{\alpha 1}$の値がゲートの非制御値，かつ$FF_{\alpha 2}$の値が出力制御値，かつ$FF_{\alpha 1}$と$FF_{\alpha 2}$の両方の制御が必要なら，リタイミング後に必ず未検出の必須故障が残ることになる．これは，$FF_{\alpha 1}$と$FF_{\alpha 2}$の値がつくる同様の信号線の値を再現するような$FF_{\beta 1}$と$FF_{\beta 2}$の値の組合せが存在しないからである．

タイプIIIリタイミングを行った後にスキャンシフト数が減るためには，以下の条件を全て満たすことが必要である．

1）スキャンシフト数が減るのはタイプIII後方リタイミングを行ったときである．

2）α構造に含まれるゲートの種類は，AND, NAND, OR, NORのいずれかである．

3）$FF_{\alpha 1}$と$FF_{\alpha 2}$は制御の必要がある．

4）$FF_{\alpha 2}$の値が出力制御値である．

5）$FF_{\alpha 1}$の値がゲートの非制御値である．

未検出故障を残さないための条件は，

6）$FF_{\alpha 1}$と$FF_{\alpha 2}$の両方の制御が必要のとき，$FF_{\alpha 1}$の値がゲートの制御値かまたは，$FF_{\alpha 2}$の値が出力制御値でない．

6．5　テストベクトルの変更とテスト系列生成

タイプIII後方リタイミングによってテスト系列が短縮するかどうかはテストベクトルに依存するため，タイプIII後方リタイミングを行う前にテストベクトルを生成する必要がある．しかし，リタイミングを行うことによって回路構造が変更するため，元のテストベクトルをそのまま用いると故障検出率が減少する場合がある．そこでリタイミング後の回路に適応したテストベクトルを得なければならない．提案方法では計算時間を節約するために，新たにテストベクトルを生成するのではなく，前のテストベクトルを変更して用いることを考える．

リタイミング後のテストベクトルでは，$FF_{\beta 1}, FF_{\beta 2}$の値を，表6.1に示すように，$FF_{\alpha 1}, FF_{\alpha 2}$の値とゲートの種類によって決める．問題となるのは，$FF_{\beta 1}, FF_{\beta 2}$の値が一意に決まらない場合で，例えば，ANDゲートで，$(FF_{\alpha 1}, FF_{\alpha 2})$が(0, 1)のときである．このときには，そのテストベクトルによって検出される故障を列挙し，$(FF_{\beta 1}, FF_{\beta 2})$を(0, X)と(1, 1)のどちらにした場合に，それらの故障が検出されるかを調べる．$(FF_{\beta 1}, FF_{\beta 2})$を(0, X)と(1, 1)のどちらにしても，未検出故障が生じるときには，その未検出故障が，他の別のテストベクトルによって検出されるかどうかを調べる．すべての故障が，変更後のテストベクトルの少なくとも一つによって検出されるように，$FF_{\beta 1}, FF_{\beta 2}$の値を決定する．

テストベクトル変更のアルゴリズムを図6.12に示す．このアルゴリズムでは，まず，$(FF_{\beta 1}, FF_{\beta 2})$の値が一意に決まるテストベクトルを見つけ，それによって検出される故障を対象から除く．その後は，残った故障の全てが検出されるように，各テストベクトルの$(FF_{\beta 1}, FF_{\beta 2})$の値を決める．計算処理は与えられたテストベクトルの順に行うが，このとき検出必要故障という概念を用いる．検出必要故障とは，そのテストベクトルでは検出されるが，以降の未処理のテストベクトルでは検出できない故障のことで，必須故障も検出必要故障に含まれる．従って，検出必要故障が必ず検出されるように$(FF_{\beta 1}, FF_{\beta 2})$の値を決める．もし，検出必要故障がなければ，より多くの未検出故障が検出されるように値を決定する．

例6.1　ANDゲートを含むα構造をタイプIII後方リタイミングによって，β構造に変換した場合について考える．4つのテストベクトルとそれが検出する故障が表6.2のように与えられたと仮定する．表6.2には，$FF_{\alpha 1}, FF_{\alpha 2}$の値と，リタイミング後に変更したときの$FF_{\beta 1}, FF_{\beta 2}$の値が示されている．例えば，テストベクトル$t_1$では，$FF_{\beta 1}, FF_{\beta 2}$の値は一意に決まり，変更後も変更前と同じ故障が検出される．一方，テストベクトルt_2では，$FF_{\beta 1}, FF_{\beta 2}$の値の候補は2通りあり，どちらを選択するかによって検出される故障が異なる．テストベクトルt_2～t_4の$FF_{\beta 1}, FF_{\beta 2}$の値を決めるときには，まず，$FF_{\beta 1}, FF_{\beta 2}$の値が一意に決まる

Procedure: Test vector modification
/* T = { $t_1, t_2, ..., t_v$ } is a set of original test vectors */
for (i = 1 to v)
　if (t_i can be changed uniquely)
　　Drop the faults which are detected by t_i from the target fault list;
for (i = 1 to v)
{
　if (t_i can be changed uniquely)
　　continue;
　if (t_i detects any necessary-to-detect fault)
　　Determine the values of $FF_{\beta 1}$ and $FF_{\beta 2}$ so that the necessary-to-detect faults are detected;
　else
　　Determine the values of $FF_{\beta 1}$ and $FF_{\beta 2}$ so that more faults are detected;
　Drop the faults which are detected by the modified test vector from the target fault list;
}

図6.12　テストベクトル変更のアルゴリズム

表6.2　テストベクトルと検出される故障

テストベクトル	検出される故障	(FFα1, FFα2)	(FFβ1, FFβ2)	変更後検出される故障
t1	f1, f2	(1, 1) →	(1, 1)	f1, f2
t2	f2, f3, f4, f5	(0, 1) →	(0, X)	f2, f3, f5
			(1, 1)	f4
t3	f1, f4, f6, f7	(0, 1) →	(0, X)	f4, f6
			(1, 1)	f1, f7
t4	f6, f7, f8	(0, 1) →	(0, X)	f6
			(1, 1)	f7, f8,

テストベクトルt_1で検出される故障を対象から除く．その結果，t_2では，故障f_3, f_4, f_5についてのみ考える．このうち，テストベクトルt_3, t_4では検出されない故障f_3が検出必要故障であり，t_2の$(FF_{\beta 1}, FF_{\beta 2})$を(0, X)に決め，$f_3$が必ず検出されるようにする．次に，$t_3$では，故障$f_4$が検出必要故障であり，それが検出されるテストベクトルを見つける．その結果，t_3の$(FF_{\beta 1}, FF_{\beta 2})$を(0, X)に決める．最後にテストベクトル$t_4$では，$t_1$〜$t_3$で検出されなかった故障$f_7, f_8$を検出するように，$(FF_{\beta 1}, FF_{\beta 2})$を(1, 1)に決める．このようにして，未検出故障が残らないようにテストベクトルを変更する．

6．6　実験結果

ISCAS '89 のベンチマーク回路[52]に対して行った実験の結果を示す．実験では，まずタイプI，タイプII リタイミングを行い，FF数を少なくした．次に，その回路に対して，文献 [69] のテスト圧縮技術を用いて，テストベクトルを生成した．次に，回路内のα構造を探して，スキャンシフト数が減る条件を満たすかどうかを判断しながら，テストベクトルを変更した．結果の比較のため，元の回路に対して，短縮スキャンシフト法を用いてテスト系列を生成した．リタイミングの効果を見るため，実験においては，スキャンチェーン上のFFの配列は与えられていると仮定し，与えられた回路のネットリストと同じ順に配列した．

表6.3の左側には，左から順に，元の回路に対する，FF数，テストベクトル数，フルスキャンシフト法によるテスト系列長，短縮スキャンシフト法によるテスト系列長，2つの系列長の比，テストベクトル生成の計算時間を含まないテスト系列生成のための計算時間が示されている．表の右側はリタイミングを行った回路に対する結果で，左から順に，FF数，テストベクトル数，フルスキャンシフト法によるテスト系列長，タイプIとタイプII リタイミングの両方を行った回路に対する短縮スキャンシフト法によるテスト系列長("短縮1"で示す)，タイプIとタイプII リタイミングの後にタイプIII 後方リタイミングを行った回路に対する短縮スキャンシフト法によるテスト系列長("短縮2"で示す)，フルスキャンシフト法によるテスト系列長に対する"短縮2"で表されるテスト系列長の比，テストベクトル変更のための計算時間を表す．使用した計算機はSun-SS/Classicである．

表6.3　実験結果

	元の回路						リタイミング					
回路	FF	v	フル	短縮	比	時間(s)	FF	v	フル	短縮1	短縮2	比	時間(s)
s5378	179	108	19619	16163	0.82	165	155	115	18095	13011	13011	0.72	0.1
s9234	226	144	32914	29936	0.91	692	199	153	30799	27277	27190	0.88	9.9
s13207	669	240	161469	138212	0.86	2712	506	236	120158	116845	116845	0.97	24.7
s15850	597	119	71759	59337	0.83	2365	580	112	65652	53393	53362	0.81	11.4
s38417	1636	112	184980	179669	0.97	24659	1584	112	179104	177346	177346	0.99	0.1

表に含まれない他のベンチマーク回路では，リタイミングを可能にする構造が含まれず，タイプI，タイプII，タイプIII リタイミングを行うことができなかった．s5378とs38417では，スキャンシフト数が減るための条件を満たすα構造が存在しなかった．またs13207では，リタイミング後にFFの値がドントケアとなるテストベクトルが1つだけ見つかったが，テスト系列長は短くならなかった．s9234とs15850において，スキャンシフト数が減るようなα構造が存在し，テスト系列長も短くなった．

6．7　まとめ

ゲートレベルで表された回路のFFの位置を移動するリタイミング技法を，短縮スキャンシフト法に応用した方法を提案した．スキャン回路のテスト系列を短くするために，FF数の減少が有効であることは，短縮スキャンシフト法以外にも共通する事実であるが，短縮スキャンシフト法では，たとえFF数が減らなくても，テスト系列長が短くなる場合がある．ここでは，2種類のリタイミングを行い，FF数を十分少なくした後に，FF数が変わらないリタイミングを行った．その際，テストベクトルに対するFFの制御の必要性を調べ，また，リタイミング後にそれらのテストベクトルが利用できるようなテストベクトル変更の方法を説明した．実験を行った結果，FF数の多い比較的大規模な回路に対してその有効性を

示すことができた．

今後は，他の種類のリタイミング，例えばもっと広範囲で行うリタイミングなどを考え，中規模，小規模の回路に対しても有効となる方法を開発する．またパーシャルスキャン回路に対しても，提案方法を適用することが考えられる．パーシャルスキャン回路においては，リタイミングによる故障検出率の変化についても考えなくてはならない．

第7章　区別系列を持つ順序回路のテスト系列生成

この章では，区別系列を持つ順序回路の設計法とテスト系列生成法[46]について述べる．まず，7.1節では，区別系列について説明する．7.2節では，区別系列を持つ順序回路に対するHennieのテスト系列生成法[23]について述べる．7.3節では，与えられた回路に対し，区別系列を求める方法を述べ，それを計算機上で実現するアルゴリズムを示す．7.4節では，順序回路を区別系列を持つように設計する方法について，状態遷移図レベルとゲートレベルの2つの記述レベルで実現する方法を述べる．7.5節では，区別系列を持つ順序回路に対して，組合せ部分のテストベクトルと区別系列を用いた，縮退故障に対する完全なテスト系列生成の方法を述べる．7.6節では，7.5節の縮退故障に対するテスト系列を，故障検出率を下げることなく，テスト系列長を短くする方法を述べる．7.7節では，7.4節の方法で設計した順序回路に対して，7.5節，7.6節の方法によりテスト系列を生成した実験の結果を述べる．最後に7.8節でまとめを述べる．

7．1　区別系列とは

区別系列とは，出力系列を観測することによって，順序回路の内部状態を特定することができるような入力系列である．例えば表7.1に示す状態遷移を行う回路では，01という入力系列が区別系列となる．初期状態q_1(00)，q_2(01)，q_3(11)，q_4(10)に対して，系列01を印加したときの出力系列は，それぞれ11，01，10，00であり，出力系列がすべて異なるため，初期状態を区別することができる．また，区別系列を印加したときには，印加した後に遷移する状態も知ることができる．もし，系列を印加した後の状態が，すべて同じ状態ならば，その系列を同期系列とよぶ．

7．2　区別系列用いたHennieのテスト系列生成法

ここでは，順序回路の同定問題として区別系列を用いてテスト系列を求めるHennieの方法[23]を説明する．これは，回路の状態遷移表が与えられたとき，その回路がその状態遷移表と同じ動作を行うこと

表7.1　状態遷移表

		次状態		出力	
入力		0	1	0	1
現状態	q_1 (00)	q_1	q_1	1	1
	q_2 (01)	q_1	q_3	0	0
	q_3 (11)	q_2	q_4	1	0
	q_4 (10)	q_3	q_2	0	1

を一意に表す入力系列を求め，この入力系列によって回路の入出力動作が状態遷移表と一致するかを調べる方法である．このテスト系列で検出することのできる故障は，状態遷移表を変化させるあらゆる故障である．ただし，故障が起こった場合でも状態数は増えないと仮定している．テスト系列は状態遷移表を基に生成されるので，回路の実装方法と独立にテスト系列は有効となる．

n個の状態 $s_1, s_2, ..., s_n$ とm個の入力ベクトル $x_1, x_2, ..., x_m$ で定義された状態遷移表と区別系列 ρ が与えられたと仮定する．まず，区別系列 ρ を印加したとき，n個の状態に対して異なる出力系列が得られるかどうか，即ち，区別系列 ρ によってn個の状態が識別できるかどうかを調べる系列を，次のように構成する．

$\alpha = \rho - \tau(q_1, s_1) - \rho - \tau(q_2, s_2) - \rho ... - \rho - \tau(q_n, s_n) - \rho$

ここで，$\tau(q_k, s_k)$は状態q_kから状態s_kへ遷移させる入力系列で，q_k は区別系列を印加した後に遷移する状態で$s_1, s_2, ..., s_n$ のいずれかである．$\rho - \tau(q_k, s_k)$ は，系列 ρ に続いて系列 $\tau(q_k, s_k)$を印加することを表す．また，回路の初期状態は既知で，それをs_1と仮定している．系列 α は，回路をある状態s_kに遷移させて，その直後に区別系列を印加することを，s_1からs_nのn個の状態に対して繰り返している．s_1からs_nの状態に対する区別系列の出力応答がすべて異なっていれば，n個の異なる状態が存在することが確かめられる．

次に，各状態に対して，m個の異なる入力ベクトルを印加したときの出力と状態遷移を確かめる．起こった状態遷移が正しいかどうかは，区別系列を印加することによって確かめることができる．状態s_kで入力x_jに対する出力と状態遷移を確かめる系列は

$L_j^k = \tau(q_j, s_{k-1}) - \rho - \tau(q_{k-1}, s_k) - x_j - \rho$

となる．L_j^kを印加する前の状態をq_jとすると，系列 $\tau(q_j, s_{k-1}) - \rho$は，先の系列 α にも現れているので，これによって確実に状態をq_jから s_{k-1}に遷移させ，それを区別系列によって確かめることができる．1つの状態 s_kに対して，すべての入力ベクトル $x_1, x_2, ..., x_m$ を確かめるためには，

$L^k = L_1^k - L_2^k - ... - L_m^k$

が必要で，さらに，$s_1, s_2, ..., s_n$のすべての状態に対して同様の系列が必要で，その系列は，

$\beta = L^1 - L^2 - L^3 - ... - L^n$

となる．

次に，αとβからなるテスト系列の系列長を評価する．系列 α 中には，区別系列がn+1個あり，状態遷移のための系列$\tau(q_k, s_k)$がn個ある．系列$\tau(q_k, s_k)$の長さは高々n-1であるので，区別系列の長さをgとすると，αの長さ$|\alpha|$は次のようになる．

$|\alpha| \le g(n+1) + n(n-1)$

また，同様にL_j^kの長さ$|L_j^k|$が，

$|L_j^k| \le 2n + 2g - 1$

となることより，系列βの長さ$|\beta|$は，

$|\beta| \le mn(2n + 2g - 1)$

となる．結局，αとβからなるテスト系列長は，

$|L| = |\alpha| + |\beta| \le g(n+1) + n(n-1) + mn(2n + 2g - 1) < 2n(m+1)(g+n)$

となる[47].

７．３　区別系列を求める方法

ここでは，与えられた状態遷移表から区別系列を求める方法[48]を述べる．まず，状態遷移表に含まれるすべての状態を含む集合Q_mをつくり，これを初期状態とする．以降は，この集合を，入力値に対して得られる出力値によって分割していく．例えば，表7.2の状態遷移表から，図7.1の第1段に示されるQ_mを求める．各状態に対して入力0を印加したときの，次状態と出力を，第2段の左側に示す．$q_2(1)$は初期状態q_1に対する次状態がq_2であることを示している．また，2つの状態の集合は，出力値が0と1の違いによって分割したものである．この例では，出力値が0であれば，初期状態がq_1またはq_2であり，出力値が1であれば，初期状態がq_3またはq_4であることが分かる．ところが，初期状態q_3とq_4に対する次状態が共にq_3であるため，以降どのような入力を加えたとしても，この2つの状態を区別することができない，即ち，区別系列の最初の入力は0ではあり得ないと分かる．最初の入力として1を印加したときの，次状態と出力値は図7.1の第2段の右側に示される．初期状態q_1に対する出力値が唯一，1であるので，状態q_1は区別できたことになる．次に0，1を加えた結果が，図7.1第3段に示される．入力1を印加したとき，第2段で区別できなかった $\{q_3(2), q_4(3), q_1(4)\}$の各状態が，出力値が0の $\{q_4(2), q_1(3)\}$と出力値1の $\{ q_1(4) \}$の集合に分割できた．最終的には，第4段に示されるように，すべての初期状態がそれぞれ1つづつの集合に分割され，各状態を出力値によって区別することができる，そのときの入力系列1 1 1が表7.2の状態遷移表に対する区別系列である．

状態遷移表が与えられていない場合の，区別系列を求めるアルゴリズムを図7.2に示す．このアルゴリズムは基本的には，前述した状態の集合を出力値によって分割していく方法と同様で，ただし，ある入力を印加したときの各状態に対する次状態と出力値を，必要に応じて計算する点が異なる．まず1行目で，すべての状態を含む集合Q_mをつくり，それが，ある入力v_jを印加したときの出力値によって分割できるかを判断する（4行目）．状態の集合の要素数がすべて1になったとき，区別系列を求められたことになる（7行目）．" MARK "以降で行われる処理は，2行目から7行目で行われる処理と同様であるが，出力値が既に計算された入力を用いて次状態を計算する点が異なる．

表7.2　状態遷移表

		次状態		出力	
入力		0	1	0	1
現状態	q_1	q_2	q_1	0	1
	q_2	q_1	q_3	0	0
	q_3	q_3	q_4	1	0
	q_4	q_3	q_1	1	0

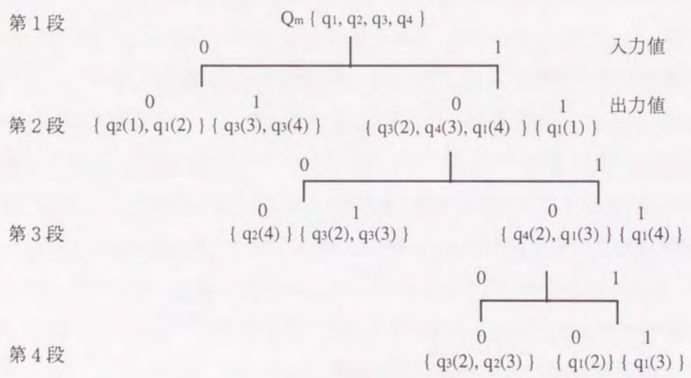


図7.1　状態の分割

７．４　区別系列を持つ順序回路の設計

７．４．１　状態遷移レベルでの設計

与えられた状態遷移表に2つの入力記号を新たに加えて状態遷移表を拡大することによって，順序回路が区別系列を持つようにする設計法[18],[19]を述べる．新たにつけ加える入力記号をそれぞれe_0, e_1とし，それぞれに対する状態遷移関数と出力関数を次のように定義する．状態割当$Y_1 Y_2 ... Y_p$に対する状態S_iに対する状態遷移関数は，

$\delta(S_i, e_0) = S_j$,

$\delta(S_i, e_1) = S_k$.

となる．ここで，S_j，S_kはそれぞれ，状態割当$0Y_1 Y_2 ... Y_{p-1}$, $1Y_1 Y_2 ... Y_{p-1}$ に相当する．また出力関数は，

$$\lambda(S_i, e_0) = \lambda(S_i, e_1) = \begin{cases} O_1 \text{ (if } Y_p = 0 \text{)} \\ O_2 \text{ (if } Y_p = 1 \text{)} \end{cases}$$

となる．もし，元の状態遷移表にS_j，S_kに対応する状態が定義されていないならば，新たにそれを付加する．

Procedure: Generation of Distinguishing Sequence

1: let Q_m be a set including all states;
2: ***for*** (every input vector v_i) {
3:　logic simulation for every state and v_i;
4:　***if*** (no state set can be partitioned) continue;
5:　calculate next state sets from each state;
6:　***if*** (the cardinality of each state set is one)
7:　　return (distinguishing sequence)
8: MARK:
9:　***for*** (every already simulated input vector v_k) {
10:　　***if*** (no state set can be partitioned) continue;
11:　　calculate next state sets from each state;
12:　　***if*** (the cardinality of each state set is one)
13:　　　return (distinguishing sequence)
14:　　go to MARK;
15: }
16: }
17: return (no distinguishing sequence)

図7.2　区別系列を求めるアルゴリズム

このように設計された順序回路をASM (Augmented State Machine) 回路という．ASM回路では，状態数をnとすると，入力e_0が連続する長さ$\lceil \log_2 n \rceil$の入力系列が区別系列となる．しかもそれは同期系列でもある．

ASM回路の設計を例を用いて説明する．表7.3(a)の状態遷移表が与えられたとする．例えば，状態q_1(00)は，先の定義により，入力e_0，e_1に対し状態(00)，(10)に遷移する．また，出力は，O_1を0，O_2を1に割り当てる．表7.3(b)が区別系列を持つように設計された結果である．そこでは，長さ2の入力系列e_0-e_0が区別系列となり，と同時に印加後の状態がすべてq_1となる同期系列でもある．

７．４．２　ゲートレベルでの設計

ゲートレベルで記述された順序回路に対し，１本の外部入力線といくつかのゲートを付加することによって，区別系列を持つような設計法を述べる．この方法は，付加した１本の外部入力線（CONTROL

表7.3　(a)元の状態遷移表と　(b)入力を付加した状態遷移表

(a)

		次状態		出力	
入力		I0	I1	I0	I1
現状態	q1 (00)	q2	q4	0	1
	q2 (01)	q1	q3	1	1
	q3 (10)	q2	q4	0	1
	q4 (11)	q4	q4	0	0

(b)

		次状態				出力			
入力		I0	I1	e0	e1	I0	I1	e0	e1
現状態	q1 (00)	q2	q4	q1	q3	0	1	0	0
	q2 (01)	q1	q3	q1	q3	1	1	1	1
	q3 (10)	q2	q4	q2	q4	0	1	0	0
	q4 (11)	q4	q4	q2	q4	0	0	1	1

線）に印加する信号値を制御することによって，内部状態の制御や観測を可能にした，スキャン回路の一表現法である．図7.3が区別系列を持つように設計された回路で，これをSAG回路 (Scan circuit with Added Gates) とよぶ．図中で黒く塗られたゲートが新たに付加されたゲートである．CONTROL線に信号値1を印加したとき，組合せ部分からの出力がFFに入力する通常動作で，信号値0を印加したとき，FFの状態によってOUT_0での出力系列が一意に決まる．つまり，SAG回路の区別系列は，FF数がnのとき，CONTROL線を0にしたときの長さnの入力系列となる．また，区別系列を印加した後の状態は，外部入力IN_0から印加された値によって一意に決まるため，この区別系列は同期系列でもある．この方法では，1つのFFに対して3つのゲートと，外部出力OUT_0で状態を観測するための3つのゲートと，CONTROL線の反転をつくるNOTゲートを付加することが必要で，必要な付加ゲートの総数は$3n$+4となる．

７．５　テスト系列生成

縮退故障を対象とし，区別系列と組合せ部分のテストベクトルからテスト系列を生成する方法を述べる．組合せ部分のテストベクトルを

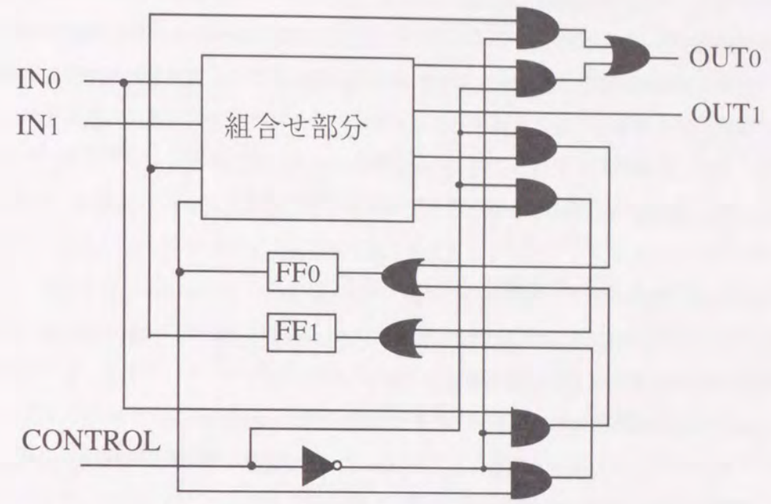


図7.3　ＳＡＧ順序回路

$T = \{ t_1(p_1, s_1), t_2(p_2, s_2), ..., t_v(p_v, s_v) \}$

とする．ここで，$t_i(p_i, s_i)$ は，状態s_iで入力ベクトルp_iを印加することを表す．ある故障 f に対して，$t_i(p_i, s_i)$が組合せ部分のテストベクトルであることは，テストベクトルt_iが故障fを顕在化し，正常時とは異なる出力または次状態をつくることを意味する．従って，テストベクトルt_iを印加した後，区別系列を印加すれば，出力を観測することによって，故障の有無を調べることができる．テストベクトルt_iで組合せ的に検出される故障は，状態をs_iにした後，入力ベクトルp_iと区別系列ρを順に印加することによって外部出力で検出することができる．与えられたテストベクトル$t_1, t_2, ..., t_v$に対するテスト系列は，

$\Lambda = I_i - t_1 - \rho - \pi(q_1, s_2) - t_2 - \rho - ... - \pi(q_{v-1}, s_v) - t_v - \rho$

となる．$\pi(q_i, s_j)$ は状態q_iを状態s_jに遷移させる入力系列で，状態q_iは区別系列を印加した後の状態である．また，I_iは初期化系列で，ここでは，7.2節で述べたα系列を用いる．同期系列の性質も持った区別系列を印加した後の状態をs_1とすると，α系列は次のようになる．

$\alpha = \rho - \rho - \pi(s_1, s_2) - \rho - \pi(s_1, s_3) - \rho - ... - \pi(s_1, s_n) - \rho$

組合せ部分のテストベクトルと区別系列から生成されたテスト系列は，系列ΛにおけるI_iを系列 $\alpha - \pi(s_1, q_1)$に置き換えた系列である．α系列の後の$\pi(s_1, q_1)$は区別系列を印加後の状態s_1をテストベクトルt_1に必要な状態q_1に遷移させるために必要である．

区別系列から構成されたα系列は，2.5節で述べた初期化が困難な回路に対しても有効である．まず，最初に区別系列を印加したとき，もし出力値が正常値と異なるなら，故障が検出できたことになる．出力値が正常値と同じで状態遷移が異なるときは，もう一度区別系列を印加することで，故障を検出する

ことができる．また，出力値と状態遷移の両方が正常値と同じときには，故障が顕在化されておらず，回路を初期化することができる．このようにα系列で，最初に二度区別系列を印加することによって，回路の状態の初期化が行える．しかも，ASM回路とSAG回路では，区別系列が同時に同期系列でもあるので，区別系列を印加した後の状態が，常に一意に決まるため，初期状態に関わらず同じ系列でテストできる．もし，区別系列を印加した後の状態が異なるなら，出力応答によって以降に印加すべき系列を選ばなくてはならないが[49]，ASM回路やSAG回路ではその必要がない．

７．６　テスト系列の短縮法

前節で述べたテスト系列Λを，故障検出率を下げることなく短縮する方法を述べる．テスト系列を短縮するために考える点は次の三点である．

１）正常時と故障時の状態を区別するために区別系列の一部を用いる．

２）テストベクトルの順序を考慮することにより，状態遷移のための系列を短縮する．

３）初期化系列として，α系列より短い系列を用いる．

７．６．１　区別系列の短縮

テスト系列Λでは，テストベクトル印加後の状態を調べるために区別系列を用いたが，故障検出のために必ずしも区別系列全体を必要としない場合がある．縮退故障のみを対象とすれば，故障が存在した場合でも，テストベクトルを印加した後の状態は，いくつかに限られる．提案方法では，その限られた故障時の状態が，区別系列の一部で区別できる場合には，不要な入力ベクトルを省略することによって，テスト系列を短縮する．

例7.1：図7.4の回路の故障fを検出する場合について説明する．この回路の区別系列は，$(I_1, I_2) = (00, 11)$で，テストベクトル$t = (I_1, I_2, FF_1, FF_2) = (X, 1, 0, X)$によって故障$f$が顕在化され，次状態の$FF_2$の値が正常値と異なる1となる．その後，区別系列を印加することによって，故障fの影響が外部出力OUTで観測される．ところが，故障シミュレーションを行うと，図7.4に示されるように，区別系列の最初のベクトル$(I_1, I_2) = (0, 1)$を印加しただけで，外部出力で故障fの影響を観測することができる．このように，テストベクトルと区別系列の一部だけで，故障を検出できる場合がある．

区別系列中の不要な入力ベクトル見つける方法を，以下に述べる．区別系列がg個の入力ベクトル$b_1, b_2, ..., b_g$で構成されているとする．テストベクトルt_iを印加し，それに続いて区別系列を印加し，同時に未検出故障を対象に故障シミュレーションを実行する．もし，入力ベクトルb_jを印加したとき故障の影響が外部出力に現れ，b_{j+1}からb_gの入力ベクトルを印加したときにはどの未検出故障の影響も外部出力に現れないなら，テストベクトルt_iの後に印加すべき区別系列を$b_1, b_2, ..., b_j$で構成しても，故障検出率は変わらない．このようにしてテストベクトル $t_i(p_i, s_i)\ (i = 1, ..., v)$と，それに続いて印加する区別系列

のうち不要な入力ベクトルを削除した部分的なテスト系列の集合 $T_s = \{ t_1(p_1, s_1) - \rho_1, t_2(p_2, s_2) - \rho_2, ..., t_v(p_v, s_v) - \rho_v \}$ が得られる．系列ρ_iは区別系列の一部または全部を表す．その後，初期化系列と，各テストベクトルに必要な状態$s_1, s_2, ...s_v$に遷移させる系列をつなげてテスト系列を構成する．

７．６．２　テストベクトルの順序

テスト系列Λでは，区別系列を印加した後の状態をテストベクトルの印加に必要な状態に遷移させる必要があったが，テストベクトルを印加する順序を考慮することによって，そのような遷移のための系列を省略したり，短くすることができる．例えば，もし，区別系列を印加した後の状態と同じ状態を必要とするテストベクトルを次に印加すべきテストベクトルとして選んだなら，遷移のための系列は不要となる．提案方法では，テストベクトルと区別系列の一部からなる系列に対して，系列印加前に必要な状態と，系列印加後に到達する状態を前もって調べておき，遷移のための系列が最も少なくなるように，逐次的にテストベクトルの順序を決める．

図7.4の回路において，すべてのゲートの出力線の0と1の縮退故障を対象に，テスト系列を生成する場合を説明する．まず，組合せ部分のテストベクトルとして，表7.4に示されるt_1～t_6のテストベクトルが生成されたとする．区別系列の一つの入力ベクトル$(I_1, I_2) = (0, 1)$をeで表すと，区別系列は$e - e$で，系列$t_i - e - e$によって故障の影響を外部出力で観測することができる．ここで，7.6.1節で述べた方法により，区別系列中の必要なベクトルを調べると，表7.5のような結果が得られる．テストベクトルt_1, t_3, t_5では区別系列が全く不要であることが分かる．また，表7.5には，各系列の印加前の状態と印加後の状態

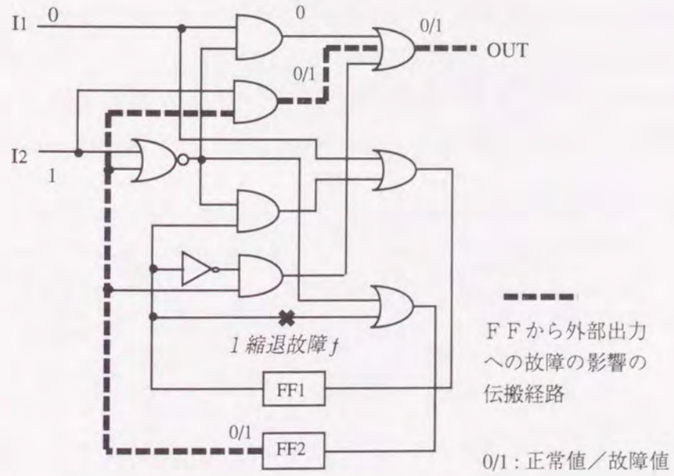


図7.4　区別系列による故障の影響の伝搬

が示されてある．これらの状態を比較しながら，テストベクトルの順序を決める．１番目のテストベクトルは，初期化系列を印加した後の状態に応じて決めるのであるが，ここでは，t_1が１番目に選ばれたとする．系列 t_1 - e - e 印加後の状態は $(FF_1, FF_2) = (0, 0)$ であり，これは，t_3の印加前に必要な状態と一致する．即ち，系列 t_1 - e - e を印加後に，t_3を印加すれば遷移のための系列を省略することができる．表7.5の各系列を連結して，

$L_s = t_1$ -e - e - t_3 - e - e - t_2 - t_4 - t_5 - t_6

なる系列L_sを生成すると，遷移のための系列を加えることなく，各系列の印加前に必要な状態を得ることができる．

７．６．３　初期化系列の短縮

初期化系列としてのα系列は，回路を未知の状態から既知の状態に遷移させることと，区別系列や遷移のための系列が故障を顕在化するかどうかを確かめる系列であった．しかし，そのためには長い系列が必要であったので，回路を既知の状態に遷移させるだけの系列を初期化系列として用いることを考える．このような初期化系列として，区別系列を２つつなげた系列 ρ - ρを用いる．この系列を用いると，7.5節で述べたように，初期状態がどのようであっても特定の既知の状態へ遷移するか，あるいは，初期化ができない場合には，故障が顕在化されそれが外部出力で観測される．

次に，α系列を用いないために起こり得る問題について述べる．提案するテスト系列を短縮する方法では，テストベクトル印加前に必要な状態に遷移する系列を適当に選んでいるが，そのために本来検出できるはずの故障が検出できなくなる場合がある．図7.5(a)の回路において故障fを検出するテストベクトルは，$(I_1, I_2, FF_0) = (0, 1, 0)$である．このテストベクトルを印加する前に，状態$FF_0$=1から状態$FF_0$=0に遷移させる入力ベクトル $(I_1, I_2) = (0, 1)$ を印加すると，図7.5(b)のように，その入力ベクトルによって故障が顕在化され，次にテストベクトルを印加したときには，故障の影響が相殺して消えてしまう．つまり，状態遷移のための入力系列によって故障が顕在化されたとき，検出されるべき故障が検出できなくなる場合がある．状態遷移のために，α系列に含まれる一部の系列を用いた場合には，このように故障が検出できなくなる場合は起こらない．なぜなら，もしそのようなことが起こるなら，α系列を印加したときにも同様のことが起こり，それは区別系列を印加することによって，外部出力において正常値と異なる値として観測できるからである．

図7.5に示すような検出できなくなる故障が生じるかどうかを調べるために，提案方法では，生成したテスト系列に対して故障シミュレーションを行う．もし，このような故障が生じるときには，その原因となっている状態遷移のための入力系列を，別の入力系列に置き換える．例えば図7.5の場合でも，$(I_1, I_2) = (1, 0)$なる入力ベクトルを用いたなら，入力ベクトル $(I_1, I_2) = (0, 1)$ と同じ状態遷移を起こし，しかも故障fが未検出にならない．万一，どのような入力系列を用いても，検出できなくなる故障が生じる場合には，初期化系列としてα系列を印加する．しかし，どのような入力系列を用いても，そのような故障が生じる場合は，非常に少ないと考えられる．

表7.4　組合せ部分のテストベクトル

	I_1	I_2	FF_1	FF_2
t_1	0	0	0	0
t_2	0	1	X	0
t_3	1	1	0	0
t_4	1	X	1	1
t_5	0	X	0	1
t_6	0	0	1	0

表7.5　テストベクトルと区別系列

	印加前状態	入力系列	印加後状態
t_{s1}	0 0	t_2 - e - e	0 0
t_{s2}	X 0	t_1	1 1
t_{s3}	0 0	t_3 - e - e	0 0
t_{s4}	1 1	t_4	X 1
t_{s5}	0 1	t_5	X 0
t_{s6}	1 0	t_6 - e - e	0 0

７．７　実験結果

MCNC[51]の状態遷移表から設計したASM回路に対する結果を表7.6に示す．表中の"PI"，"FF"，"ゲート"は，外部入力線数，FF数，ゲート数である．"DSの系列長"，"ベクトル数"の欄はそれぞれ，区別系列の長さ，組合せ部分のテストベクトル数を表している．"Λの系列長"は，7.5節で述べた，α系列，テストベクトル，状態遷移のための入力系列から構成されたテスト系列長で，テスト系列の短縮法

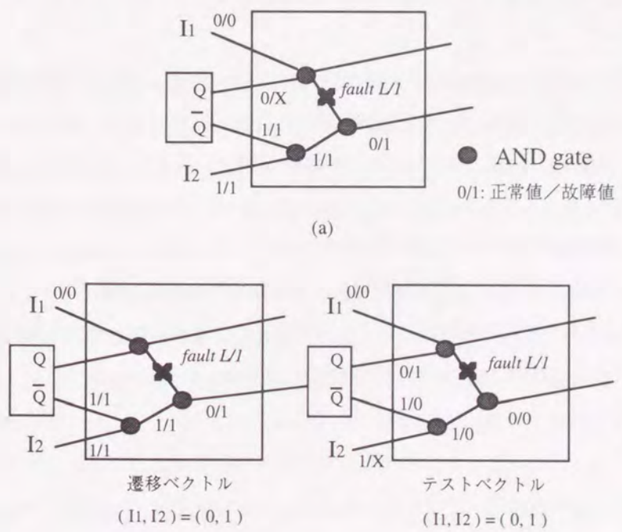


図 7.5 (a) 組合せ部分のテスト生成と (b) 問題のある遷移系列

表7.6　ＡＳＭの順序回路に対する実験結果

回路	PI	FF	ゲート	DSの系列長	ベクトル数	Λの系列長	Λsの系列長	[23]の系列長	fsim 時間 (sec)	total 時間 (sec)
dk14	4	3	100	3	27	192	114	334	10.5	12.1
dk15	4	2	66	2	23	106	69	136	3.8	4.7
dk16	3	5	208	5	57	769	387	1472	83.7	90.1
dk17	3	3	69	3	21	155	89	160	6.7	7.6
dk27	2	3	41	3	17	140	91	176	3.4	3.8
dk512	2	4	88	4	29	329	173	448	16.3	17.8

を用いない場合である．"Λsの系列長"は，7.6節で述べた方法でテスト系列を短縮した場合である．いずれのテスト系列によっても，すべての回路に対して100%の故障検出率が得られた．比較のため，文献[23]の方法による，区別系列を用いたテスト系列の長さを"[23]の系列長"の欄に示す．これらの結果より，提案方法によりテスト系列が十分短縮されたことが分かる．"fsim時間"と"total時間"の欄はそれぞれ，故障シミュレーションとテスト系列生成全体にかかった計算時間を表す．使用した計算機は，Sun-SS/Classicである．故障シミュレーションアルゴリズムとしては，文献[53]のPROOFSアルゴリズムに基づくものを用いた．この結果より，計算時間の大部分が故障シミュレーションに費やされたことが分かる．

ISCAS'89のベンチマーク回路[52]からSAG回路を設計し，それに対してテスト系列を生成した結果を表7.7に示す．表中には，元の回路のゲート数とSAG回路の設計のために付加されたゲート数が示されている．"Λsの系列長"は表7.6と同様，テスト系列の短縮法を用いたときのテスト系列長である．この表には示されていないが，もしこれらの回路に対してα系列を用いたテスト系列を生成すると，これらの回路はFF数nに対して状態数が2^nと非常に膨大となり，そのテスト系列も非常に長くなる．これらの全ての回路においても，検出可能な縮退故障に対して，100%の故障検出率が得られた．

実験の結果，ASM，SAGどちらの回路に対しても，提案方法によって短いテスト系列を得ることができた．しかも，いずれの回路においても故障検出率は100%であった．

７．８　まとめ

区別系列を持つ順序回路の設計法とテスト系列生成法について述べた．区別系列を用いた順序回路のテストは以前から研究されているが，テスト系列長が非常に長くなるという欠点があった．提案方法は，この欠点を克服し短いテスト系列で，しかも従来と同様の高い故障検出率を実現する方法である．与えられた回路の区別系列を求めることは，一般的には容易でないため，ここでは，状態遷移表レベル

表7.7　ＳＡＧの順序回路に対する実験結果

回路	PI	FF	元のゲート	付加ゲート	DSの系列長	ベクトル数	Λsの系列長	検出不能	fsim 時間 (sec)	total 時間 (sec)
ms27	5	3	13	13	3	9	39	1	0.5	0.8
ms208	12	8	104	28	8	29	261	3	17.6	22.2
ms298	4	14	131	46	14	26	393	0	45.6	53.9
ms344	10	15	173	49	15	18	292	0	33.9	42.3
ms382	4	21	177	67	21	30	673	0	108	122
ms386	8	6	164	22	6	70	493	0	37.4	53.3
ms420	20	16	212	52	16	47	799	5	89.0	105
ms444	4	21	202	67	21	29	652	22	126	142
ms526	4	21	214	67	21	56	1233	1	161	188
ms820	19	5	292	19	5	97	587	0	177	238
ms832	19	5	290	19	5	97	586	15	192	255
ms953	17	29	424	85	29	85	2551	1	635	729
ms1196	15	18	545	58	18	120	2285	0	500	627
ms1494	9	6	651	22	6	102	715	13	299	444

とゲートレベルの２つの設計法を説明した．テスト系列を生成する際の特徴の１つは，組合せ部分のテストベクトルを利用したことで，これにより，高い故障検出率でしかも短いテスト系列長を実現することができた．また，区別系列全体が必要かどうかを，故障シミュレーションを行うことによって詳細に調べたり，初期化系列や状態遷移のための入力系列が短くなるような方法も提案した．実験を行った結果は，提案方法の有効性を示している．

今後の課題としては，対象故障を縮退故障だけでなく，他の種類の故障も含めたもっと広い範囲の故障を対象に，テスト系列の短縮法を考える必要がある．区別系列は本来，状態遷移関数や出力関数を変えるようなどのような故障も検出できるような性質を持っているため，縮退故障以外の故障に対しても適用可能と考えられる．また，与えられた回路の区別系列を高速に見つける方法の開発や，少ない付加回路で区別系列を持つ回路を設計する方法の開発が望まれる．

第8章　結論

一般社会で広くコンピュータが利用されるにつれ，その信頼性を高める意味からも，論理回路のテストの重要性は益々高まっている．テストに関するこれまでの研究を見ると，組合せ回路を対象にした場合と比較して，順序回路を対象にした場合には，依然として解決困難な問題が多く残されている．本論文では，順序回路をテストする際に起こる問題を取り上げ，それを解決するための方法を提案した．ここでの目的の一つは，テスト系列長を短縮することである．実用的な計算時間で順序回路に対して高い故障検出率を得ることは，一般的には困難であるが，テスト容易化設計法を導入することによって，高い故障検出率を得ることも容易になる．しかし，欠点として，テスト系列が長くなるという問題がある．そこで，高い故障検出率を維持したままで，テスト系列を短縮する方法を提案した．

提案した方法の一つは，スキャン回路に対するテスト系列を短縮するための，短縮スキャンシフト法である．スキャン回路は，現在最も広く用いられているテスト容易化設計法の一つである，スキャン設計によって得られた回路で，順序回路に対するテスト生成の問題が，組合せ回路に対するテスト生成の問題に帰着し，その結果，高い故障検出率を達成することが容易になる．しかし，テストベクトル数とFF数に比例してテスト系列が長くなるという欠点がある．短縮スキャンシフト法は，テスト系列が長くなる原因のスキャンシフト操作での入力系列をできるだけ短くする方法である．スキャンシフト操作は，生成されたテストベクトルに基づいて，FFを制御や観測するための操作で，フルスキャンシフト法とよぶ従来法では，常に全てのFFの制御と観測を行っていた．しかし，各テストベクトル毎に検出する故障を決めたときには，一部のFFを制御や観測するだけで故障を検出することができる．ただし，このとき問題となるのは，各テストベクトルで検出すべき故障をどのように選ぶかである．対象故障を多くしたときには，スキャンシフト操作での制御や観測の必要なFFを多く見積もり過ぎ，逆に対象故障を少なくしたときには，テスト系列を生成したときに多くの未検出故障を残す．短縮スキャンシフト法で対象とした必須故障は，一種の検出困難な故障であるため，生成したテスト系列で高い故障検出率を達成することができた．また，必須故障は数が少ないため，スキャンシフト操作で必要な入力系列を短くすることができた．

短縮スキャンシフト法の基本的な概念は単純であるため，これを様々な回路や他のテスト技法へ適用した．フルスキャン回路に対する短縮スキャンシフト法では，スキャンチェーン上のFFの配列とテストベクトルを印加する順序を工夫することによって，短いテスト系列長を実現した．スキャンチェーン上のFFの配列は，テストベクトルに対して計算された各FFの制御や観測の必要性に基づき，頻繁に制御の必要のあるFFをスキャン入力に近く，頻繁に観測の必要のあるFFをスキャン出力に近くになるように配列した．テストベクトルの順序を決める際には，制御のみ，または観測のみのスキャンシフト操作が少なくなるように，即ち，必要な制御と観測のスキャンシフト操作の長さが同じくらいのテストベ

クトルが連続するようにした．さらに，回路の状態遷移を利用してテストベクトルの順序を決める方法も提案した．ベンチマーク回路に対して実験を行った結果は，100%の故障検出率を達成し，テスト系列長では従来法に比べて最大で約1/10にまで短縮することができた．

短縮スキャンシフト法をパーシャルスキャン回路に対して適用したときには，テスト系列を短くするだけでなく，故障検出率が高くなるような方法を用いた．初期化系列を生成すること，非スキャンFFの状態と比較しながらテストベクトルの順序を決めること，順序回路に対するテスト生成アルゴリズムを用いたことは，いずれも故障検出率を高くするためである．テスト系列の短縮のためには，FFの制御や観測の必要性に基づいた，スキャンFFの選択と，スキャンチェーン上のFFの配列を考えた．実験の結果では，FFの配列やテストベクトルの順序決めの効果が詳しく調べられ，提案方法の有効性が示された．また，他者の研究結果と比較しても，提案方法は短いテスト系列で高い故障検出率が得られることが示された．本研究ではスキャンFF数は予め与えられると仮定したが，スキャンFF数を少なくするための方法については，さらに開発の必要がある．

リタイミングを応用した短縮スキャンシフト法についても研究を行った．リタイミングはゲートレベルの回路上で，出力関数を変えずにFFの位置を変える技法で，これを用いることによってFF数を減らすことができる．従来のフルスキャンシフト法を用いたときには，テスト系列を短縮するために，FF数を減らすだけで十分であったが，短縮スキャンシフト法を用いたときには，FF数が減らない場合でもテスト系列が短くなるようなリタイミングを行う必要がある．本論文では，FF数が変わらないようなリタイミングを行ったときに，テスト系列が短くなるかどうかについて，組合せ部分のテストベクトルを用いて調べる方法を提案し，どのような条件を満たしたとき，テスト系列が短くなるかを示した．また，リタイミングの結果得られた回路に対するテストベクトルを，元のテストベクトルから，故障検出率を下げることなく変更する方法を提案した．実験の結果は，故障検出率を維持したまま，テスト系列を短縮できることを示しており，同時に，短い計算時間でテストベクトルが変更できることを示している．

フルスキャン回路に対する方法では，いずれも100%の故障検出率が得られたわけであるが，パーシャルスキャン回路に対しては，さらに故障検出率を高くする要求が残る．パーシャルスキャン回路に対する故障検出率を高くする問題は，本質的に，スキャン設計を行わない順序回路の故障検出率を高くすることと共通する．理想的には，どのようなテスト容易化も行わない順序回路のテストを考えなければならないが，それに至る前段階としては，スキャン設計よりも面積オーバーヘッドが少なく，かつ，回路の性能の低下の少ないテスト容易化設計法を開発する必要がある．

本研究で提案したもう一つの方法は，区別系列を持つ順序回路に対するテスト系列の短縮法である．区別系列は，順序回路の内部状態を識別するための入力系列であり，これを利用することによって，高い故障検出率を達成するテスト系列が容易に生成できる．しかし従来法では，そのようなテスト系列は非常に長くなるという問題がある．この問題を解決するために，順序回路の組合せ部分に対するテスト

ベクトルと区別系列を用いたテスト系列生成法を提案した．提案方法では，対象故障を，一般的によく用いられる縮退故障とすることで，組合せ回路に対するテストベクトル生成のアルゴリズムを利用することができ，その結果，テスト系列を大幅に短縮することができた．また区別系列の特性を利用することによる，初期化系列の短縮や，テストベクトルを印加した後に必要となる区別系列の一部を省略することによっても，テスト系列の短縮を実現した．与えられた順序回路の区別系列を得る方法としては，最初に計算によって区別系列を見つけることを試み，それに失敗したときには，状態遷移表レベルまたはゲートレベルで区別系列を持つように回路を設計した．いくつかのベンチマーク回路に対して実験を行った結果は，状態遷移表レベルとゲートレベルのいずれの設計の回路に対しても，従来よりも短いテスト系列で100%の故障検出率を得ることができた．

また，本研究で対象とした故障モデルは縮退故障のみであるが，今後は遅延故障など，縮退故障以外の故障に対するテスト系列を短縮する方法を開発する必要がある．その際にも，本論文で提案した様々な技法が応用できると考えられる．

謝辞

本研究は，大阪大学大学院工学研究科応用物理学専攻において，樹下行三教授の御指導のもとで行ったものである．本研究を遂行するにあたり，終始御指導賜り，また，有益な議論および御助言頂きました樹下行三教授に心より感謝致します．

本論文の作成に関し，詳細な御検討，貴重な御教示を頂きました大阪大学大学院工学研究科応用物理学専攻 豊田順一教授，伊東一良教授に深く感謝致します．同じく本論文の作成において貴重な御教示を頂きました大阪大学大学院工学研究科応用物理学専攻 興地斐男教授，増原宏教授，志水隆一教授，河田聡教授，中島信一教授，八木厚志教授，石井博昭教授，後藤誠一教授，岩崎裕教授，および同研究科物質・生命工学専攻 一岡芳樹教授，川上則雄教授に深く感謝致します．

大阪大学大学院工学研究科応用物理学専攻 梶原誠司博士には，本研究を遂行するにあたり有益な議論，貴重な御助言頂きました．また，本論文作成に際し，詳細な御検討，御指導頂きました．梶原誠司博士に心より感謝致します．

大阪大学大学院工学研究科応用物理学専攻 小松雅治助教授には，本研究を遂行するにあたり有益な議論，貴重な御助言を頂きました．小松雅治助教授に深く感謝致します．また同専攻板崎徳禎博士には，本研究において有益な議論，貴重な御助言を頂き，特に，計算機実験を行う際に懇切な御指導頂きました．板崎徳禎博士に深く感謝致します．

広島市立大学情報科学部助手上田祐彰氏には，プログラム実装及び計算機実験において御指導，御助言頂きました．上田祐彰氏に深く感謝致します．

樹下研究室の諸氏には一方ならぬ御支援，御協力頂きました．ここに記して感謝致します．

参考文献

[1] H. Fujiwara: Logic Testing and Design for Testability, MIT Press Series in Computer Systems, The MIT Press, Cambridge, Massachusetts, London, England,1985.

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

[3] 樹下行三，浅田邦博，唐津修：VLSIの設計II，岩波書店，1985.

[4] V. D. Agrawal and S. C. Seth: Test Generation for VLSI Chips, IEEE Computer Society Press, Washington D. C., 1988.

[5] T. W. Wiiliams and K. P. Parker, "Design for Testability - A Survey," Proc. IEEE, vol. 71, no. 1, pp. 311-325, 1983.

[6] J. P. Roth, "Diagnosis of Automata Failure: A Calculus and a Method," IBM J. Res. Develop., pp. 278-291, July 1966.

[7] P.Goel : "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," IEEE Trans. Comput., vol. C-30, no. 3, pp.215-222, Mar. 1981.

[8] H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms," IEEE Trans. on Comput., vol.C-32, pp. 1137-1144, Dec. 1983.

[9] Y. Takamatsu and K. Kinoshita, "CONT: A Concurrent Test Generation Algorithm," Proc. Int'l Symp on Fault-Tolerant Computing, pp. 22-27, June 1986

[10] M.H.Schulz, E.Trischeler and T.M.Sarfert, "SOCRATES:A Highly Efficient Automatic Test Pattern Generation Systems," IEEE Trans. on CAD., vol. 7, pp. 126-137, Jan. 1988.

[11] Y. Matsunaga and M. Fujita, "A Fast Test Pattern Generation for Large Scale Circuits," Proc. Synthesis and Simulation Meeting and Int'l Interchange, pp. 263-271, Apr. 1992.

[12] A. Tamama and N. Kuji, "Integrating an Electron-beam System into VLSI Fault Diagnosis", IEEE Design & Test of Comput., vol. 3, no. 4, pp.23-29. Aug. 1986.

[13] E.Wolfgang : "Electron Beam Testing", in "Handbook of Advanced Semiconductor Technology and Computer Systems", (ed.G.Rabaat) Van Norstand, Reinhold Co. 1987.

[14] 古川康男，稲垣雄史，" LSIの非接触診断技術，" 電学論C, vol. 107, No. 3, pp. 245-250.

[15] N. K. Jha and S. Kundu: Testing and Reliable Design of CMOS Circuits, Kluwer Academic Publishers, 1990.

[16] S. Devadas, H. T. Ma and A. R. Newton, "A Synthesis and Optimization Procedure for Fully and Easily Testable Sequential Machines, IEEE Trans. on CAD, vol. 8, no. 10, pp.1100-1107, Oct. 1989.

[17] I. Pomeranz and S. M. Reddy, "Design and Synthesis for Testability of Synchronous Sequential Circuits Based on Strong-Connectivity," Proc. Int'l Symp. Fault-Tolerant Comput., pp. 492-501, June 1993.

[18] H. Fujiwara, Y. Nagao, T. Sasao and K. Kinoshita, "Easily Testable Sequential Machines with Extra Inputs", IEEE Trans. on Comput., vol. C-24, no. 8, pp.821-826, 1975.

[19] S. Shibatani and K. Kinoshita, "Synthesis of Easily Testable Sequential Circuits with Checking Sequences", Proc. First Asian Test Symp., pp. 200-205, Nov. 1992.

[20] N. J. Y.Williams, J. B. Angell, "Enhancing Testability of Large Scale Integrated Circuits via Test Points and Additional Logic," IEEE Trans. on Comput., vol. C-22, no. 1, pp.46-60, 1973.

[21] A. Toth and C. Holt, "Automated Data Base Driven Digital Testing," IEEE Trans. Comput., vol. C-23, pp.13-

19, Jan. 1974.

[22] E. B. Eichelberger, T. W. Williams, "A Logic Design Structure for LSI Testability," Proc. 14th Design Automation Conf., pp.462-468, 1977.

[23] F. C. Hennie, "Fault-Detecting Experiments for Sequential Circuits", Proc. 5th Annual Symp. on Switching Circuit Theory and Logic Design, pp.95-110, 1964.

[24] S. Malik, E. M. Sentovich, R. K. Brayton and A. Sangiovanni-Vincentelli, "Retiming and Resynthesis: Optimizing Sequential Networks with Combinational Techniques", IEEE Trans. on CAD, vol. 10, pp. 74-84, Jan. 1991.

[25] S. Lejmi, B. Kaminska and E. Wagneur, "Resynthesis and Retiming of Synchronous Sequential Circuits", Proc. Int'l Symp. on Circuits and Systems, pp. 1674-1677, 1993.

[26] De Micheli, "Synchronous Logic Synthesis: Algorithms for Cycle-Time Minimization", IEEE Trans. on CAD, vol. 10, pp. 63-73, Jan. 1991.

[27] S. Dey and S. T. Chakradhar, "Retiming Sequential Circuits to Enhance Testability", Proc. VLSI Test Symp., pp. 28-33, May 1994.

[28] D. Kagaris and S. Tragoudas, "Partial Scan with Retiming", Proc. Design Automation Conf., pp. 249-254, June 1993.

[29] S. B. Akers, "Binary Decision Diagrams," IEEE Trans. on Comput., vol. C-27, no. 6, pp. 509-516, June 1978.

[30] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," IEEE Trans. on Comput., vol. C-35, no. 8, pp. 677-691, Aug. 1986.

[31] 石浦菜岐佐，" BDDとは，" 情報処理学会誌, vol. 34, no. 5, pp. 585-592, May 1993.

[32] T. Geewala, "CrossCheck: A Cell Based VLSI Testability Solution," Proc. Design Automation Conf., pp. 706-709, June 1989.

[33] C. Timoc, M. Buehler, T. Griswold, C. Pina, F. Scott and L. Hess, "Logical Models of Physical Failures," Proc. Int'l Test Conf., pp. 546-553, Oct. 1983.

[34] D. B. Armstrong, " A deductive Method for Simulating Faults in Logic Circuits," IEEE Trans. on Comput., vol. C-21. no. 5, pp.464-471, May 1972.

[35] E. G. Ulrich and T. Baker, "The Concurrent Simulation of Nearly Identical Digital Networks," 10th Design Automation Workshop, vol. 6, pp. 145-150, June 1973.

[36] P. Goel, H. Lichaa, T. E. Rosser, T. J. Stroh and E. B. Eichelberger, "LSSD Fault Simulation Using Conjunctive Combinational and Sequential Methods," Proc. ITC, pp371-376, Nov. 1980.

[37] W.-T. Cheng and M.-L. Yu, "Differential Fault Simulation - A Fast Method Using Minimal Memory," Proc. Design Automation Conf., pp. 424-428, June 1989.

[38] S Seshu, "On an Improved Diagnosis Program," IEEE Trans. on Electron. Comput., vol. EC-14, pp. 76-79, Feb. 1965.

[39] N. Gouders and R. Kaibel, "PARIS: A Parallel Pattern Fault Simulation for Synchronous Sequential Circuits," Proc. Int'l Conf. on CAD, pp. 542-545, Nov. 1985.

[40] H. Kubo, "A Procedure for Generating Test Sequences to Detect Sequential Circuit Failures," NEC Res. Dev., no. 12, pp. 69-78, Oct. 1968.

[41] G. R. Putzolu and J. P. Roth, "A Heuristic Algorithm for the Testing of Asynchronous Circuits," IEEE Trans. on Comput., vol. C-20, pp. 639-647, June 1971.

[42] P . Muth, "A Nine-Valued Circuit Model for Test Generation," IEEE Trans. on Comput., vol. C-25, pp.630-636, June 1976.

[43] E. Auth and M. H. Shulz, "A Test-Pattern-Generation Algorithm for Sequential Circuits," IEEE Design & Test of Comput., vol. 8, no. 2, pp.72-85, June 1991.

[44] A. Ghosh, S. Devadas and A. R. Newton, "Test Generation and Verification for Highly Sequential Circuits," IEEE Trans. on CAD, vol.10, no. 5, May 1991.

[45] T. P. Kelsey, K. K. Saluja and S. Y. Lee, "An Efficient Algorithm for Sequential Circuit Test Generation," IEEE Trans. on Comput., vol. 42, pp. 1361-1371, Nov. 1993.

[46] Y. Higami, S. Kajihara and K. Kinoshita, "Test Sequence Generation for Sequential Circuits with Distinguishing Sequences," IEICE Trans. on Fundamentals, vol. E76-A, no. 10, Oct. 1993.

[47] 樹下行三，尾崎弘：ディジタル代数学，共立出版 1966.

[48] 樹下行三，藤原秀雄：ディジタル回路の故障診断（上），工学図書 1983.

[49] A. Gill, "State Identification Experiments in Finite Automata", Info. and Control, vol.4, 1961.

[50] M. Abramovici and P. S. Parikh, "WARNING: 100% Fault Coverage May Be Misleading!!", Proc. Int'l Test Conf., pp. 662-668, Oct. 1992.

[51] R. Lisanke, "FSM Benchmark Suite", Microelectronics Center of North Carolina, Research Triangle Park, North Carolina, 1987.

[52] F.Brglez, D. Bryan and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," Proc. Int'l Symp. on Circuits and Systems, pp. 1929-1934, 1989.

[53] T. M. Niermann, Wu-Tung Cheng and J. H. Patel, "PROOFS: A Fast, Memory Efficient Sequential Circuit Fault Simulator", Proc. Design Automation Conf., pp.535-540, June 1990.

[54] Y. Higami, S. Kajihara and K. Kinoshita, "Reduced Scan Shift: A New Testing Method for Sequential Circuits," Proc. Int'l Test Conf, pp. 624-630, Oct. 1994.

[55] Y. Higami, S. Kajihara and K. Kinoshita, "A Reduced Scan Shift Method for Sequential Circuit Testing," IEICE Trans. on Fundamentals, vol. E77-A, no. 12, Dec. 1994.

[56] P.- C. Chen, B.-D. Liu and J.-F. Wang, "Overall Consideration of Scan Design and Test Generation," Proc. Int'l Conf. on CAD, pp. 9-12, Nov. 1992.

[57] S. P. Morley and R. A. Marlett, "Selectable Length Partial Scan: A Method to Reduce Vector Length" Proc. Int'l Test Conf., pp. 385-392, Oct. 1991.

[58] H. Fujiwara and A. Yamamoto, "Parity-Scan Design to Reduce the Cost of Test Application," IEEE Trans. on CAD, vol. 12, no. 10, pp. 1604-1611, Oct. 1993.

[59] R. Gupta and M. A. Breuer, "Ordering Storage Elements in a Single Scan Chain," Proc. Int'l Conf. on CAD, pp. 408-411, Nov. 1991.

[60] S. Narayanan, R. Gupta and M. Breuer, "Configuring Multiple Scan Chains for Minimum Test Time," Proc. Int'l Conf. on CAD, pp. 4-8, Nov. 1992.

[61] S. Y. Lee and K. K. Saluja, "An Algorithm to Reduce Test Application Time in Full Scan Designs" Proc. Int'l Conf. on CAD, pp. 17-20, Nov. 1992.

[62] S. Y. Lee and K. K. Saluja, "Sequential Test Generation with Reduced Test Clocks for Partial Scan Designs," Proc. IEEE VLSI Test Symp., pp. 220- 225, May 1994.

[63] W.-J. Lai, C.-P. Kung and C.-S. Lin, "Test Time Reduction in Scan Designed Circuits," Proc. European Design Conf., pp. 489-493, Mar. 1993.

[64] J.-S. Chang and C.-S. Lin, "A Test Clock Reduction Method for Scan-Designed Circuits," Proc. Int'l Test Conf, pp. 331-339, Oct. 1994.

[65] H. Higuchi, K. Hamaguchi and S. Yajima, "Compact Test Sequences for Scan-Based Sequential Circuits," IEICE Trans. on Fundamentals, vol. E76-A, no. 10, pp. 1676-1683, Oct. 1993.

[66] Y. Bertrand, F. Bance and M. Renovell, "Multiconfiguration Technique to Reduce Test Duration for Sequential Circuits," Proc. Int'l Test Conf., pp. 989-997, Oct. 1993.

[67] J. -S. Chang and C. -S. Lin, "Test Compaction for Combinational Circuits," Proc. Asian Test Symp., pp. 20-25, Nov. 1992.

[68] I. Pomeranz, L. N. Reddy and S. M. Reddy, "COMPACTEST: A Method to Generate Compact Test Sets for Combinational Circuits," Proc. Int'l Test Conf., pp. 194-203, Oct. 1991.

[69] S. Kajihara, I. Pomeranz, K. Kinoshita and S. M. Reddy, "Cost-Effective Generation of Minimal Test Sets for Stuck-at Faults in Combinational Logic Circuits", Proc. Design Automation Conf., pp. 102-106, June 1993.

[70] Y. Higami, S. Kajihara and K. Kinoshita, "A Partial Scan Algorithm Based on Reduced Scan Shift," Proc. Asian Test Symp., pp. 336-341, Nov. 1994.

[71] Y. Higami, S. Kajihara and K. Kinoshita, "Partial Scan Design and Test Sequence Generation Based on Reduced Scan Shift Method," Journal of Electronic Testing: Theory and Applications, vol. 7, nos. 1/2, pp. 115-124, Aug./Oct. 1995.

[72] E. Trischler, "Incomplete Scan Path with an Automatic Test Generation Methodology," Proc. Int'l Test Conf., pp. 153-162, Nov. 1908.

[73] V. D. Agrawal, K.-T. Cheng, D. D. Johnson and T. Lin, "Designing Circuits with Partial Scan," IEEE Design and Test of Comput., vol. 5, pp. 8-15, Apr. 1988.

[74] K.-T. Cheng and V. D. Agrawal, "A Partial Scan Method for Sequential Circuits with Feedback," IEEE Trans. on Comput., vol. 39, pp. 544-548, Apr. 1990.

[75] K. S. Kim and C. R. Kime, "Partial Scan by Use of Empirical Testability," Proc. Int'l Conf. on CAD, pp. 314-317, Nov. 1990.

[76] M. Abramovici and J, J. Kulikowski, "The Best Flip-Flops to Scan," Proc. Int'l Test Conf., pp. 166-173, Oct. 1991.

[77] V. Chickermane and J. H. Patel, "A Fault Oriented Partial Scan Design Approach," Proc. Int'l Conf. on CAD, pp. 400-403, Nov. 1991.

[78] 梶原誠司，板崎徳禎，樹下行三，"可観測な環境を前提としたテストパターン生成," 信学論D-I, vol. J73 D-1, no. 2, pp. 342-349, Mar. 1990.

[79] T.P.Kelsey, K.K.Saluja and S. Y. Lee, "An Efficient Algorithm for Sequential Circuit Test Generation," IEEE Trans. on Comput., Vol. 42, No. 11, pp. 1361-1371, Nov. 1993.

[80] Y. Higami, S. Kajihara and K. Kinoshita, "Test Sequence Compaction by Reduced Scan Shift and Retiming," Proc. Asian Test Symp., pp. 169-175, Nov. 1995.

[81] S. M. Reddy and R. Dandapani, "Scan Design Using Standard Flip-Flops", IEEE Design & Test of Comput., vol. 4, pp.52-54, 1987.

[82] B. Vinnakota and N. K. Jha, "Synthesis of Sequential Circuits for Parallel Scan", European Conf. on CAD,

pp.366-370, 1992.

[83] S.Mallela and S.Wu, "A Sequential Circuit Test Generation System," Proc. Int'l Test Conf., IEEE Comput. Society Press, Los Alamitos, Calif., pp.57-61, 1985.

[84] W.Cheng, "The Back Algorithm for Sequential Test Generation," Proc. Int'l Conf. Comput. Design, IEEE Comput. Society Press, Los Alamitos, Calif., pp.66-69, 1988.

[85] R.Marlett, "An Effective Test Generation System for Sequential Circuits," Proc. Design Automation Conf., pp.250-256, June 1986.

[86] T. Ogihara, S. Saruyama and S. Murai, " Test Generation for Sequential Circuits Using Individual Initial Value Propagation", Proc. Int'l Conf. on CAD, pp.424-427, Nov. 1988.

[87] D. Ho Lee and S. M. Reddy, "A New Test Generation Method for Sequential Circuits," Proc. Int'l Conf. on CAD, pp. 446-449, Nov. 1991.

発表論文一覧

論文誌発表論文

1．Yoshinobu Higami, Seiji Kajihara, and Kozo Kinoshita
"Test Sequence Generation for Sequential Circuits with Distinguishing Sequences"
IEICE Trans. on Fundamentals, vol. E76-A, no.10, pp. 1730-1737, Oct. 1993.

2．Yoshinobu Higami, Seiji Kajihara, and Kozo Kinoshita
"A Reduced Scan Shift Method for Sequential Circuit Testing"
IEICE Trans. on Fundamentals, vol. E77-A, no. 12, pp. 2010-2016, Dec. 1994.

3．Yoshinobu Higami, Seiji Kajihara, and Kozo Kinoshita
"Partial Scan Design and Test Sequence Generation Based on Reduced Scan Shift Method"
Journal of Electronic Testing: Theory and Applications, vol. 7, nos. 1/2, pp. 115-124, Aug./Oct. 1995.

国際会議発表論文

1．Yoshinobu Higami, Seiji Kajihara, and Kozo Kinoshita
"Reduced Scan Shift: A New Testing Method for Sequential Circuits"
Proc. IEEE International Test Conference, pp. 624-630, Oct. 1994.

2．Yoshinobu Higami, Seiji Kajihara, and Kozo Kinoshita
"A Partial Scan Algorithm Based on Reduced Scan Shift"
Proc. IEEE Asian Test Symposium, pp. 336-341, Nov. 1994.

3．Yoshinobu Higami, Seiji Kajihara, and Kozo Kinoshita
"Test Sequence Compaction by Reduced Scan Shift and Retiming"
Proc. IEEE Asian Test Symposium, pp. 169-175, Nov. 1995.

研究会発表論文

1．樋上喜信，梶原誠司，樹下行三
”短縮スキャンシフトによる順序回路のテスト”
信学技報ICD研, ICD93-150, pp. 21-28, Dec. 1993.

2．樋上喜信，梶原誠司，樹下行三
”状態遷移を用いた短縮スキャンシフトによる順序回路のテスト”
信学技報VLD研, VLD93-109, pp. 87-94, Mar. 1994.

3．樋上喜信，梶原誠司，樹下行三
”短縮スキャンシフトによるパーシャルスキャンアルゴリズム”
情報処理学会ＤＡシンポジウム論文集, pp. 107-112, Aug. 1994.