2006－ＡＲＣ－168（９）
2006／6／8

プロセッサの命令レベル自己テストのためのテスト容易化設計

中里　昌人†　　大竹　哲史†　　井上美智子†　　藤原　秀雄†

† 奈良先端科学技術大学院大学情報科学研究科　〒630–0192 けいはんな学研都市
E-mail: †{masato-n,ohtake,kounoe,fujiwara}@is.naist.jp

あらまし　本稿では，テンプレートを利用したテストプログラム生成法から合成された任意のテストプログラム実行時に起こる誤りマスクを回避するプロセッサのテスト容易化設計法を提案する．テンプレートを利用したテストプログラム生成法では，モジュール単体テスト生成で検出可能な故障が，テストプログラム実行時では検出されない誤りマスクが問題である．提案手法は，誤りマスクを完全に除去するという意味で，テンプレートレベル故障検出効率100%を達成可能である．また，提案手法は，信号線の観測のみで誤りマスクを回避するため，遅延オーバーヘッドがなく実動作速度テストが可能である．

キーワード　命令レベル自己テスト，テスト容易化設計，プロセッサ，誤りマスク，テストプログラムテンプレート

Design for Testability of Software-Based Self-Test for Processors

Masato NAKAZATO†, Satoshi OHTAKE†, Michiko INOUE†, and Hideo FUJIWARA†

† Graduate School of Information Science, Nara Institute of Science and Technology
Kansai Science City, 630–0192, Japan
E-mail: †{masato-n,ohtake,kounoe,fujiwara}@is.naist.jp

Abstract　In this paper, we propose a design for testability method for test programs of software-based self-test using test program templates. Software-based self-test using templates has a problem of error masking where some faults detected in a test generation for a module are not detected by the test program synthesized from the test. The proposed method achieves 100% template level fault efficiency in a sense that the proposed method completely resolves the problem of error masking. Moreover, the proposed method adds only observation points to the original design, it enables at-speed testing and does not induce delay overhead.

Key words　software-based self-test, design for testability, processor, error mask, test program template

1. まえがき

近年，VLSI技術の発展により高集積および高性能なプロセッサが開発される一方で，プロセッサのテストは困難な問題になっている．一般に，プロセッサのテスト方式としてスキャン設計に基づくテスト方式が用いられる．しかし，スキャン設計を適用したプロセッサには，面積オーバーヘッド，遅延オーバーヘッド，消費電力が大きいおよび実動作速度によるテストが困難であるといった問題点がある．一方，テストパタン生成器，応答解析器を回路に組み込んだ自己テスト方式は，実動作速度でのテストが可能である．しかし，擬似ランダムパタンの印加によりテスト実行時の消費電力がプロセッサの通常動作時の消費電力よりも増加するという問題がある．

これらのテスト方式に対して，プロセッサの機能である命令を利用する自己テスト方式が注目されている．命令レベル自己テストでは，プロセッサが持つ命令を用いてテストを行う．プロセッサのテストを行うためのテストパタンは，テストプログラムとして外部テスタからプロセッサのメモリに読み込まれる．プロセッサは，テストプログラムを実動作速度で実行し，テスト応答はメモリに書き込まれる．メモリの値を外部テスタに読み出し，期待値と比較することで故障を検出する．この手法は，プロセッサの通常動作のための機能だけを用いてテストするため，ハードウェアオーバーヘッド及び遅延オーバーヘッドがなく，プロセッサの通常動作時で許容される消費電力でのテスト実行が可能である．

プロセッサの命令レベル自己テストのための方法が数多く提案されている [1-7] が，階層テスト生成に基づくテストプログラム生成法が高い故障検出率を得ることができると知られている [3-7]．階層テスト生成に基づくテストプログラム生成は，プロセッサの構成要素（モジュールと呼ぶ）に対するテスト生成（モジュール単体テスト生成と呼ぶ）とモジュール単体テスト生成で得られたテストパタンをモジュールに正当化し，テスト

応答を観測するための命令列の生成（テストプログラム合成）からなる．しかし，こられらの手法では，合成されたテストプログラムのテストパタンの正当化，テスト応答の観測のための命令列は，故障の無いプロセッサの動作のみ考慮しているため，モジュール単体テスト生成で検出できる故障が，テストプログラム実行時には検出できない（誤りマスクと呼ぶ）場合がある．

本稿では，任意のテストプログラム実行時の誤りマスクを回避するためのテスト容易化設計（DFT: Design for Testability）を提案する．提案手法は，観測点だけを追加するため，実時間テストを可能にし，ゲート挿入などによる遅延オーバーヘッドを引き起こさないという特徴を持つ．また，本稿では，テンプレートを利用して生成された任意のテストプログラムに対し，誤りマスクを回避するために十分条件を示し，観測点の挿入数最小化のための発見的手法を提案する．

以下第2節では，テスト対象となるプロセッサのモデルを定義し，第3節でテンプレートを用いたテストプログラム生成法について述べる．第4節では，プロセッサで生じる誤りマスクを述べ，第5節で誤りマスクを回避する命令レベル自己テストのためのテスト容易化設計法を示し，第6節でまとめる．

2. プロセッサモデル

図1に，本稿で取り扱うプロセッサモデルを示す．プロセッサのメモリには，バスとその制御信号，メモリへの書き込み及び読み出し制御信号が接続されている．また，プロセッサは，クロックやリセットなどの外部より制御可能な信号線を持つ．プロセッサは，メモリから命令を取り出し実行する．プロセッサは，実行する命令によってメモリからデータを読み出しまたはメモリへデータの書き込みを行う．

プロセッサは，レジスタ転送レベル（RTL: Register Transfer Level）記述を用いて設計され，ALUやマルチプレクサ等の組合せモジュール，コントローラ等の順序モジュールとそれらを接続する接続信号線（RTL信号線と呼ぶ），およびバスからなる．トライステート・バッファで実現されるバスを，トライステート・バスと呼ぶ．トライステート・バスからの出力には，ハイインピーダンス値の伝搬を抑制するために出力値を論理値（0または1）にマスクする回路[11]があるとする．正常な（故障のない）プロセッサに対して，トライステート・バスの入出力に関して以下を仮定する．

（1） 同じトライステート・バスに接続する複数の入力が活性化（トライステート・バスの入出力が接続する）されることはない

（2） トライステート・バスのどの入力も活性化していない場合，そのバスの出力値は論理値にマスクされる

プロセッサは，ALUやマルチプレクサなど，RTL記述のモジュールの階層は保存して合成され，この合成によって得られたゲートレベル記述のモジュールがテスト対象となる．提案手法は，階層化されたモジュールで構成されたプロセッサに対して適用できる．

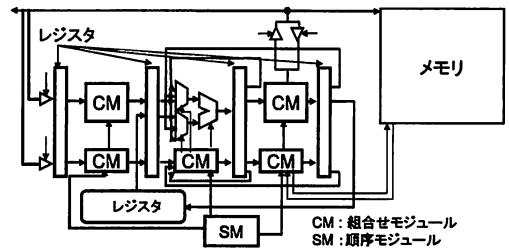

図1 プロセッサモデルの例

3. テンプレートを用いたテストプログラム生成

テンプレートを用いたテストプログラム生成では，個々のモジュール単体に対するテスト生成（モジュール単体テスト生成）とモジュール単体に対するテストパタンの正当化，および，テスト応答の観測を行うための命令列生成（テストプログラム合成）を行う．モジュール単体テスト生成で生成されたテストパタンからテストプログラムを合成することを保証するために，テンプレートを用いる．テンプレートとは，オペランドが未決定のテストプログラムで，テスト対象モジュールに対し，テストパタンの正当化及びテスト応答の観測を行うためのデーターフローを表す．図2に，テンプレートの例を示す．テンプレートは，正当化命令列，テスト命令列，観測命令列の3つの命令列で構成される．正当化命令列は，テスト対象モジュールの入力に隣接するレジスタにテストパタンを正当化する．テスト命令列は，テスト対象モジュールにテストパタンを印加し，プロセッサ中のレジスタまたは外部出力にテスト応答を伝搬する．観測命令列は，プロセッサ中のレジスタに取り込まれたテスト応答を外部出力へ伝搬する．テンプレートを用いて合成されたテストプログラムは，プロセッサが正常に動作するとき，テスト対象モジュールの入力へテストパタンを正当化し，その応答を観測可能である．

テンプレートに対して，テスト対象モジュールの入力空間，出力空間に関する制限を求め，その制限をテスト生成の制約としてモジュール単体テスト生成を行う．Kambe[5]らは，これらの入出力空間に対する制限を制約回路として抽出する手法を提案している．モジュール単体テスト生成では，テスト対象モジュールに制約回路を接続した回路に対してテスト生成を行う．制約回路の入力は，プロセッサの外部入力であるので，モジュール単体テスト生成で得られたテストパタンの値から直接テンプレートのオペランドを容易に求められるため，テンプレートからテストプログラムを容易に合成できる．

4. 誤りマスク

本節では，本稿で扱う誤りマスクと誤りマスクの評価に用いるテンプレート故障検出効率について説明する．

4.1 テンプレートレベル故障検出効率

テンプレートを用いたテストプログラム生成法では，正当化命令列，テスト命令列，観測命令列をプロセッサの正常時の動

LHI	op1	op2		
ADD.I	op4	op1	op3	
LHI	op6	op5		
ADD.I	op8	op6	op7	正当化命令列
LHI	op10	op9		
ADD.I	op12	op10	op11	
LHI	op14	op13		
ADD.I	op16	op15	op14	
LW	op17	op8(op4)		
ADD	op18	op16	op12	テスト命令列
SUB	op19	op20	op21	
LHI	op23	op22		
ADD.I	op25	op23	op24	
LHI	op27	op26		観測命令列
ADD.I	op29	op27	op28	
SW	op19	op25(op29)		

op# : 未決定のオペランド

図 2 テンプレートの例

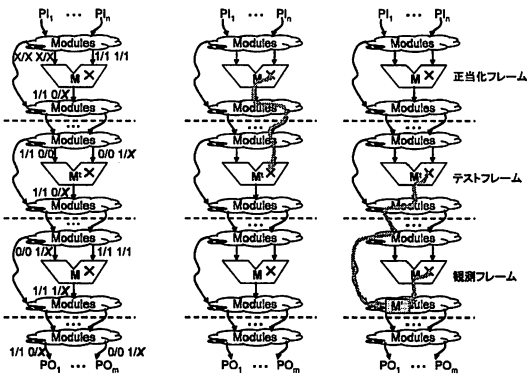

図 3 誤りマスクの例 : (a) 任意の RTL 信号線に不定値が伝搬する場合, (b) 誤りがテスト対象モジュールを通過する場合, (c) 複数の RTL 信号線に伝搬した誤りが再収斂する場合

作のみを考慮して生成している．しかし，正当化命令列，テスト命令列，観測命令列でテスト対象モジュールを利用する場合，正当化命令列，テスト命令列，観測命令列の実行時にも誤りが生じる場合がある．このとき，テストプログラムは，モジュール単体テスト生成で得られたテストパタンをテスト対象モジュールへ正当化，印加し，そのテスト応答を観測することが保証できない．これは，モジュール単体テスト生成で検出可能と判明した故障の一部がテストプログラム実行時では検出されない可能性を示している．この現象を誤りマスクと呼ぶ．本稿では，誤りマスクに対する評価尺度としてテンプレートレベル故障検出効率（FE_T: Template Level Fault Efficiency）を定義する．

テンプレート故障検出効率（FE_T）:

$$FE_T = \frac{\#TP}{\#MT}$$

ここで，#MT はモジュール単体テスト生成で検出される故障数，#TP は，#MT のうちテストプログラムでも検出される故障数とする．テンプレートレベル故障検出効率 100%を完全テンプレートレベル故障検出効率といい，誤りマスクが無いことを示す．

4.2 誤りマスクの原因

図 3 に，テストプログラム実行時のプロセッサの時間展開モデルを用いた誤りマスクの例を示す．時間展開モデルは，プロセッサの組合せ回路部分であるフレームを複数個連結した組合せ回路であり，各フレームは 1 時刻に対応する．時間展開モデル中で，モジュール単体テスト生成で生成されたテストパタンをテスト対象モジュールに印加するフレームをテストフレームと呼び，テストフレームより前の時刻に対応するフレームを正当化フレーム，テストフレームより後の時刻に対応するフレームを観測フレームと呼ぶ．

誤りマスクが起きる原因として以下の状況が考えられる．図 3 (a) に，テスト対象モジュールから不定値が伝搬する例を示す．信号線の幅は 2 ビットであり，信号線の値は “故障がない回路の信号線の値/故障がある回路の信号線の値” で表す．テスト対象モジュール M の入力に不定値が存在するとき，故障によって不定値が伝搬され，テストパタンが正当化できないこと，

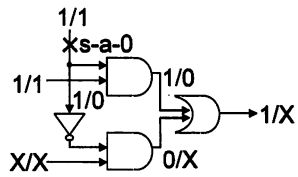

図 4 不定値伝搬の例

テスト応答が伝搬できないことがある．例えば，テスト対象モジュールが図 4 の場合，テスト対象モジュールに故障がない場合，テスト対象モジュールの出力の値は，“1” となるが，故障がある場合は，“X” となる．

図 3 (b) に，テスト対象モジュール M を含む閉路が存在するときに起こる誤りマスクの例を示す．正当化フレームで M の故障が活性化され，M の出力に現れた誤りが，テストフレームの M の入力に到達することがある．テストフレームで M に到達した値は，テストパタンとは異なる値が到達しているため M の故障を活性化できない場合がある．

図 3 (c) は，テスト対象モジュール M から複数の RTL 信号線に伝搬した誤りが再収斂するときに起こる誤りマスクの例を示す．観測フレームで M を利用して M からある外部出力へ誤りを伝搬するとき，正当化フレームまたは観測フレームにおいて M の故障が活性化することで，あるモジュール M' に複数の経路を介して誤りが伝搬し，M' の出力に誤りが伝搬しない場合がある．

M を始点とし，M' を終点とする n 本の再収斂経路の集合を n-再収斂構造と呼ぶ．また，n-再収斂構造に含まれる全ての経路があるモジュールから終点 M' までの経路を共有するとする．そのようなモジュールの中で，n-再収斂構造中のうち M' から最も遠いモジュールを，その n-再収斂構造の再収斂点と呼ぶ．

5. 命令レベル自己テストのためのテスト容易化設計

本節では，誤りマスクを回避するための DFT を提案する．

5.1 問題の定式化

提案手法は，プロセッサの全てのレジスタを初期化可能にし，さらにいくつかの RTL 信号線を可観測にすることで，全ての誤りマスクを回避する．命令レベル自己テストでは，テストプログラムを実行することにより実動作速度テストを可能にすることを目的としているので，RTL 信号線の値の観測も実動作速度で行う必要がある．そこで，RTL 信号線を実動作速度で動作可能な MISR（Multiple Input Signature Register）に接続することで観測する．提案手法は，RTL 信号線の観測のみの設計変更であるため，遅延オーバーヘッドが無く実動作速度テストが可能である．

誤りマスクを回避するためのテスト容易化設計問題を以下のように定式化する．

入力： RTL 記述プロセッサ

出力： 完全テンプレートレベル故障検出効率を達成する RTL 記述プロセッサ

最適化目標： 観測する RTL 信号線のビット数の総和の最小化

5.2 完全テンプレート故障検出効率を達成するための十分条件

テストプログラムに対して以下の仮定をおく．

(1) メモリ参照を行うアドレスに格納された値は既知であるとし，正常動作時にメモリから不定値がプロセッサに入力されることはないものとする．

(2) 値を格納するメモリアドレスには，予め，格納される期待値と異なる値に初期化されているとする．

(3) 所定のアドレスに値が格納されない又は期待値と異なる値が格納されたとき故障が検出されるものとする．所定のアドレスから値を読み込まずにテストプログラムが実行される場合も，結果として所定のアドレスに値が格納されない，または期待値と異なる値が格納されると考え故障が検出されるものとする．

提案手法が完全テンプレート故障検出効率を達成することを保証するために，任意のテストプログラム実行において誤りマスクが起きないためのプロセッサの十分条件を示す．

定理：プロセッサが以下の 3 つの条件を満たすとき，任意のテストプログラム実行において誤りマスクが起きない．

(1) 全てのレジスタが初期化可能，トライステート・バスの制御信号線とトライステート・バスのマスク回路の制御信号線を観測可能

(2) 各閉路に対して，閉路上の RTL 信号線のうち，少なくとも 1 箇所を観測可能

(3) 各 n-再収斂構造に対して，n-1 本の各再収斂経路上の RTL 信号のうち，少なくとも 1 箇所を観測可能

証明：

モジュール単体テスト生成で検出可能な任意の縮退故障 f に対するテストプログラムの実行を考える．故障 f の存在するモジュールを M とする．

まず，テストプログラム実行時に，故障を検出する，または，任意の信号線の値は不定値とならないことを示す．

故障を検出しない場合，メモリから正常動作時と同じアドレスの値を読み込む．よって，プロセッサの外部から不定値を読み込むことはない．また，故障を検出しない場合，バスに接続するトライステート・バッファの制御信号線には誤りが伝搬しておらず，そのバスに接続する信号線がバスから値を読み込むときは正常動作時にもバスから値を読み込んでいる．このとき，トライステート・バスのどの入力も活性化していなければ，そのバスの出力値は論理値（0 または 1）にマスクされるので，正常動作時にはバスの値をマスクする回路からバスに値が転送される．マスク回路の制御信号線にも誤りが伝搬していないため，故障時にも同様のデータ転送が行われる．よって，次時刻の各レジスタの値は，現在のレジスタ，外部入力，およびバス非接続時のバスに接続された RTL 信号線の値によって決定する．条件（1）より，全てのレジスタは初期化可能であるので，テストプログラムの開始時に全レジスタを初期化することにより，テストプログラムの実行中どのレジスタの値も不定値ではなくなる．

正常動作時において，テストプログラムが時刻 t にモジュール M の入力にテストパタンを正当化するとする．正当化命令列とテスト命令列を実行することで，時刻 t に M の入力に f のテストパタンが到達するか，または，時刻 t までに f が検出されることを示す．

時刻 t に M の入力に f のテストパタンが到達しない場合を考える．正常動作時にこのテストパタンを正当化するために，特定の値（0 または 1）の値をとるレジスタのビットを考える．M の入力に f のテストパタンが到達しないので，これらのビットのうちのあるビット b には正常値と異なる値が伝搬されている．テストプログラムの実行中には，レジスタは不定値を取らないので，b の値は誤りである．誤りを発生するのは M のみであるため，b を通る M の出力から M の入力までの経路 P が存在し，この経路上に誤りが伝搬している．条件（2）より，各閉路に対し，その経路上の少なくとも 1 つの RTL 信号線は観測可能であるので，P 上の少なくとも 1 つの信号線は観測可能であり，故障が検出される．よって，時刻 t に M の入力に f のテストパタンが到達するか，または，時刻 t までに f が検出される．

時刻 t に M の入力に f のテストパタンが到達する場合，故障 f が活性化され，誤りが M の出力に現れる．正常動作時には，時刻 t での M の出力は外部出力で観測可能である．よって，M からある外部出力まで誤りを伝搬する経路（誤り伝搬経路と呼ぶ）が存在する．故障が検出されないと仮定する．このとき，誤りが外部出力まで伝搬しない．誤りが外部出力まで伝搬しない場合，誤り伝搬経路上で誤りが消失するモジュール M' が存在する．あるモジュールが内部に故障を持たず，また，誤り伝搬経路上の入力にのみ誤りが伝搬しているとき，誤りはそのモジュールの出力に伝搬される．よって，M' は M 自身であるか，または，M' の誤り伝搬経路上にない入力にも誤り

が伝搬している．$M' = M$ のとき，閉路上に誤りが伝搬されるので，条件（2）より誤りが観測され，故障が検出される．M' の誤り伝搬経路上にない入力にも誤りが伝搬しているとき，M を始点とし M' を終点とする2-再収斂構造の全ての再収斂経路上に誤りが伝搬している．条件（3）より，そのうち1本の再収斂経路上の1つの RTL 信号線は観測可能であるので，誤りが観測され，故障が検出される．

以上より，モジュール単体テスト生成で検出可能な任意の縮退故障 f に対するテストプログラムで検出可能である．

□

5.3 アルゴリズム

プロセッサが 5.2 節で提案した誤りマスクを回避する3つの十分条件を満たすためのテスト容易化設計アルゴリズムを提案する．提案手法は，以下の5つのステップからなる．

ステップ1： プロセッサ中の全てのレジスタに初期化機能を追加

ステップ2： 回路グラフを作成

ステップ3： マルチプレクサの制御信号線にあたる回路グラフの辺を削除

ステップ4： 閉路上の RTL 信号線の可観測化

ステップ5： 再収斂構造上の RTL 信号線の可観測化

各ステップについて以下に説明する．

ステップ1：

プロセッサ中の全てのレジスタに初期化機能を追加する．この初期化機能は，外部入力から制御可能で，テストプログラムの開始時にレジスタの初期値を設定する．一般に，テストプログラムの開始時にはプロセッサの外部からの制御を必要とするので，レジスタの初期化機能に必要な外部入力は，テストプログラムの開始のために利用される外部入力と共有できると考え，新たな外部入力は必要としない．

ステップ2：

プロセッサの RTL 記述から回路グラフを生成する．回路グラフとは，組合せモジュール，順序モジュール，レジスタ，外部入力及び外部出力を頂点とし，各モジュール間を接続する RTL 信号線を辺とするグラフである．回路グラフの辺は重みを持ち，その重みは，RTL 信号線のビット幅とする．

ステップ3：

プロセッサ中の全てのマルチプレクサの制御信号線に対応する回路グラフの辺を削除する．ここで回路グラフの辺を削除するとは，対応する RTL 信号線を観測することを意味する．マルチプレクサが再収斂点となる再収斂構造では，マルチプレクサの複数のデータ入力線に誤りが伝搬しても誤りマスクを生じることはない．よって，条件（3）に代えて，その制御信号線を観測すれば，マルチプレクサが再収斂点となる再収斂構造に関わる誤りマスクは生じることはない．一般に，マルチプレクサで再収斂する経路について，マルチプレクサのデータ入力に到達する経路の RTL 信号線のビット数は，マルチプレクサの制御信号入力に到達する経路の RTL 信号線のビット数より大きい．これより，マルチプレクサのデータ入力に到達する再収斂経路の RTL 信号線を観測するより，マルチプレクサの制御信号入力に接続する RTL 信号線を観測することでハードウェアオーバーヘッドを削減できる．

ステップ4：

各閉路上の RTL 信号線を少なくとも1つ含み，ビット幅の総和が小さい RTL 信号線の集合を求めるために，以下の4つのステップを実行する．

ステップ4.1：辺の重みの和が最小となる閉路 C_n を選択する [8]．C_n に含まれる回路グラフの辺のうち，辺の重みが最小の辺 e_i を回路グラフから削除し，集合 E_r に加える．また，選択した閉路 C_n を集合 C_{min} に加える．この処理を，回路グラフが無閉路グラフになるまで繰り返す．

ステップ4.2：C_{min} 中の閉路を構成する辺のうち，その辺を含む閉路の数が最大な辺で重みが最小なものを e_s とする．E_{cut} に e_s を追加し，E_r の要素に e_s がある場合は，E_r の要素から e_s を削除する．また，E_r の要素に e_s が無い場合は，回路グラフから e_s を削除する．e_s を含む閉路 C_n を C_{min} から削除する．この処理を，C_{min} が空となるまで繰り返す．このとき，E_r に辺が残っている場合は，その辺を回路グラフに追加する．

ステップ4.3：回路グラフが無閉路になるまで，ステップ4.1，4.2を繰り返す．

ステップ4.4：ステップ4.1〜4.3までの処理で求めた閉路を切る辺の集合 E_{cut} の中には，回路グラフ中の全ての閉路を切るために必要以上の辺が存在している可能性がある．そこで，E_{cut} 中の各辺 e_i に対し，e_i を回路グラフに追加したとき，回路グラフが無閉路のままであるならば，E_{cut} から e_i を削除し，回路グラフに e_i を追加する．

ステップ5：

再収斂点がマルチプレクサではない各 n-再収斂構造の少なくとも n-1 本の再収斂経路に対し，少なくとも1つの RTL 信号を含み，ビット幅の総和の小さい RTL 信号線を求めるために，以下の3つのステップを実行する．

ステップ5.1：回路グラフの任意の頂点対 v_i，v_j $(v_i \neq v_j)$ について，v_j がマルチプレクサでなければ，v_i を始点とし，v_j を終点とする経路を全て取り出し，それらの経路の集合を P_{ij} とする．

ステップ5.2：P_{ij} の任意の2つの経路対 p_ℓ，p_m について，どちらか一方の経路に含まれる辺の中で重みが最小の辺を e_s とする．e_s を含む経路を P_{ij} から削除し，e_s を E_{cut} に加え，e_s を回路グラフから削除する．P_{ij} の要素が一つになるまで，この処理を繰り返す．

ステップ5.3：回路グラフの全ての頂点について，ステップ5.1，5.2を繰り返す．

これより，E_{cut} の辺に対応する RTL 信号線を観測する．提案手法では，こられの観測する RTL 信号線を MISR に入力する．

6. 実験結果

非パイプラインプロセッサとパイプラインプロセッサを対象として，提案するテスト容易化設計法の評価実験を行った．非パイプラインプロセッサは，SAYEH [9] を用い，パイプライン

表 1　観測ビット数と面積オーバーヘッド

プロセッサ名	DFT の種類	回路面積			面積オーバーヘッド (%)
		DFT 適用前の回路面積	観測ビット数	DFT による付加回路の面積	
SAYEH	完全スキャン	12389	165	1485	11.99
	提案手法		106	3132	25.28
Dlx_N	完全スキャン	58696	1379	13635	23.23
	提案手法		268	7772	13.24

プロセッサは，文献[10] を元にして作成した 5 パイプラインステージ Dlx_N を用いた．本実験では，メモリのアドレスバスを可観測とし，故障検出率の評価を行った．また，メモリからプロセッサへの制御信号線，プロセッサの外部入力は，可制御とした．

表 1 は，SAYEH と Dlx_N に対して提案するテスト容易化設計を適用した結果の観測ビット数と面積オーバーヘッドを示す．"観測ビット数" は，完全スキャンの場合は，スキャンフリップフロップの数，提案手法の場合は，MISR への入力数を示す．提案手法による観測ビット数は，両方のプロセッサに対して完全スキャンより小さくなった．提案手法による Dlx_N の面積オーバーヘッドは 13.24%と，完全スキャンより小さくなった．SAYEH の面積オーバーヘッドは 25.28%と，完全スキャンより大きくなった．これは，SAYEH では配線を最適化するために閉路が多くなり，閉路上の RTL 信号線の観測ビット数が大きくなるためである．

7. ま と め

本稿では，プロセッサの命令レベル自己テストのためのテスト容易化設計を提案した．提案手法は，プロセッサが 3 つの条件，(1) 全てのレジスタが初期化可能，トライステート・バスの制御信号線とトライステート・バスのマスク回路の制御信号線を観測可能，(2) 各閉路に対して，閉路上の RTL 信号線のうち，少なくとも 1 箇所を観測可能，(3) 各 n-再収斂構造に対して，n-1 本の各再収斂経路上の RTL 信号のうち，少なくとも 1 箇所を観測可能，を満たすことで，完全テンプレートレベル故障検出効率が達成できることを証明した．実験結果より，パイプラインプロセッサの場合，提案手法を適用したときの面積オーバーヘッドは，完全スキャンより小さくなった．非パイプラインプロセッサの場合，提案手法を適用したときの面積オーバーヘッドは，完全スキャンより大きくなった．しかし，提案手法は，観測点のみを追加するので，遅延オーバーヘッドがなく，MISR によるテスト応答の観測をするため実動作速度テストが可能である．

謝辞　本研究に際し，多くの貴重なご意見を頂きました奈良先端科学技術大学院大学の米田友和助手をはじめコンピュータ設計学講座の諸氏に深く感謝致します．本研究は一部，半導体理工学研究センター (STARC) との共同研究，２１世紀 COE プログラム（研究拠点形成費補助金），及び，日本学術振興会科学技術研究費補助金・基盤研究 B(2)(課題番号 15300018)，基盤研究 C(課題番号 18500038)，若手研究 (B)(課題番号 17700062) の研究助成による．

文　献

[1] W. -C. Lai, A. Krtic and K. -T. Cheng, "Test program synthesis for path delay faults in microprocessor cores," *Proc. International Test Conference 2000*, pp. 1080-1089, 2000.

[2] W. -C. Lai, A. Krtic and K. -T. Cheng, "Instruction-Level DFT for testing processor and IP cores in system-on-a chip," *Proc. of the Design Automation Conference 2001*, pp. 59-64, 2001.

[3] L. Chen and S. Dey, "Software-based self-testing methodology for processor cores," *IEEE Trans. on CAD,*, vol. 20, no. 3, pp. 369-380, 2001.

[4] L. Chen, S. Rabi, A. Raghunath and S. Dey, "A scalable software-based self-test methodology for programmable processors," *Proc. of Design Automation Conference 2003*, pp 548-553, 2003.

[5] K. Kambe, M. Inoue and H. Fujiwara, "Efficient template generation for instruction-based self-test of processor cores," *IEEE 13th Asian Test Symposium (ATS'04)*, pp. 152-157, 2004.

[6] 横山 真也, 神戸 和子, 井上 美智子, 藤原 秀雄, "パイプラインプロセッサ自己テストのための命令テンプレート生成," 信学技報 (DC2004-58), Vol. 104, No. 478, pp. 61-66, DEC. 2004.

[7] M. Inoue, K. Kambe, N. Hoashi, and H. Fujiwara, "Instruction-Based Self-Test for sequential modules in processors," *IEEE 5th Workshop on RTL and High Level Testing (WRTLT'04)*, pp. 109-114, 2004.

[8] M. Näher, "LEDA," Cambridge university press, 1999.

[9] Z. Navabi,"VHDL Analysis and Modeling of Digital Systems," McGraw-Hill, 1997.

[10] David A. Patterson and John L. Hennessy, "コンピュータの構成と設計 第 2 版（下）," 日経 BP 社, 1999.

[11] STARC, "RTL 設計スタイルガイド VHDL 編 初版," STARC, 2000.