コントローラ拡大とテストポイントを用いたテスト圧縮効率向上のためのテスト容易化設計

武田俊[†1]　大崎直也[†2]　細川利典[†3]　山崎紘史[†3]　吉村正義[†4]

VLSI の設計フローとの適合性とテストポイント挿入箇所の探索時間の削減のために，テストパターン数を削減するためのレジスタ転送レベルでのテストポイント挿入法が要求されている．本論文ではスキャンテストにおいて，できる限り多数の演算器の並列テストを可能にするためのレジスタ転送レベルテスト容易化設計法を提案する．提案するテスト容易化設計法はコントローラ拡大とテストポイント挿入を用いてデータパスの各演算器の入力と出力に入力テストレジスタと出力テストレジスタをそれぞれ割当てる．効率的な演算器の並列テストを可能とすることで，テスト圧縮の効率を高める．高位レベルのベンチマーク回路の実験結果は平均 6.5%の面積オーバーヘッドでテストパターン数を平均 20%削減したことを示す．

A Design for Testability Method to Improve Test Compaction Efficiency Using Controller Augmentation and Test Point Insertion

SHUN TAKEDA[†1]　NAOYA OHSAKI[†2]　TOSHINORI HOSOKAWA[†3]
HIROSHI YAMAZAKI[†4]　MASAYOSHI YOSHIMURA[†5]

Test point insertion methods to reduce the number of test patterns at register transfer level are required for the adaptability of traditional VLSI design flows and the reduction of time to search test point locations. In this paper, we propose a design-for-testability method at register transfer level to enable operational units as many as possible to be tested in parallel on scan testing. Using controller augmentation and test point insertion, the proposed design-for testability method assigns input test registers and an output test register to inputs and an output of each operational unit in a data path, respectively. Test compaction efficiency becomes high by enabling effective parallel testing for operational units. Experimental results on high-level benchmark circuits show that our proposed method reduced the number of test patterns by 20% with 6.5 % area overhead on average.

1. はじめに

近年，超大規模集積回路(Very Large Scale Integrated Circuits : VLSI)のテストコスト増大に伴い，テストパターン数削減手法が重要視されている．テストパターン数削減手法にはテスト圧縮法[1-2]やテストパターン数削減のためのテストポイント挿入[3-6]があり，多くの故障を並列にテストするテスト並列化によってテストパターン数の削減をおこなっている．

しかしながら，テストポイント挿入を用いたゲートレベルにおけるテスト並列化のためのテスト容易化設計手法(Design-for-Testability:DFT)[3-6]はゲート数が膨大であるためテストポイントの探索範囲も非常に膨大である．そのため，テスト並列化に膨大な時間を要する．また，ゲートレベルで DFT をおこなうと論理合成後の論理の変更により，遅延の増加や論理合成で実行したタイミングの最適性を損失する可能性がある．以上の理由から，ゲートレベルに変換される前の抽象度の高いレジスタ転送レベル(Register Transfer Level : RTL)の段階でテスト並列化を考慮することが重要である．

†1 日本大学大学院　生産工学研究科
Graduate School of Industrial Technology, Nihon University
†2 東京工業大学　工学院
School of Engineering, Tokyo Institute of Technology
†3 日本大学　生産工学部
College of Industrial Technology, Nihon University
†4 京都産業大学　コンピュータ理工学部
College of Faculty of Computer Science and Engineering, Nihon University

RTL でのテスト並列化手法のための DFT には階層テスト[7]に基づくものと，階層を維持しないで設計された回路を対象とするスキャンテストに基づくものが存在する．階層テスト法は演算器階層をゲートレベルでも維持することを前提として設計された回路を対象とする．そのため，回路に対する論理の最適化がなされておらず回路面積や遅延時間の増大という問題点と，階層を維持する設計は標準の VLSI の設計フローにそぐわないという問題点もある．また，階層テスト法で得られたテストプランと演算器のテストパターンを用いて階層を維持しない回路に対してテスト生成をおこなった場合，そのテストプランに基づくテストパターンでは故障検出率が低下する可能性がある．なぜなら回路の論理が変更されるためと考えられる．よって，標準の設計フローに沿った階層を維持しない演算器の並列テストのための DFT 手法が重要となる．

階層を維持しない RTL での演算器の並列テスト手法としてテストパターン数削減のための RTL テストポイント挿入法[8]が存在する．この手法はフルスキャン設計[9-10]を適用した回路の全演算器に対して同時に並列テストをおこなう．その際，テスト専用のマルチプレクサ(MUX)とレジスタの追加，1 状態のコントローラ拡大をおこう．しかしながら，回路面積オーバーヘッドが大きいという問題と 1 状態のコントローラ拡大ではテスト用の状態をすべて定義できない可能性があるという問題が存在する．そのため，本論文では回路面積の大きいテスト専用のレジスタの追加

をおこなわず，複数状態のコントローラ拡大を可能とする演算器の並列テストのための DFT 手法を提案する．

2. 諸定義

2.1 テストポイント

一般的にテストポイントは，テスト対象回路のテスタビリティ向上を図る目的で用いられる[4].

本論文では, 制御点のテストポイントとして 2 入力 MUX を用いる．これを疑似外部入力である既存のレジスタに対して接続する．任意の信号線に 2 入力 MUX を経由してレジスタへのパスを追加することで，接続したレジスタからその信号線の値を直接制御可能である．同様に，観測点としても 2 入力 MUX を使用し，これを疑似外部出力である既存のレジスタに対して接続する．任意の信号線からレジスタへのパスを追加することで，その信号線の値をレジスタで直接観測可能である．

2.2 演算器のテストレジスタ

本論文で対象とする RTL 回路はデータパスとコントローラから構成されるとする．RTL データパス回路中の演算器 j が他のレジスタを介さずに入力方向もしくは出力方向に到達可能なレジスタを演算器 j のレジスタと定義する．ここで，演算器 j の入力から入力方向に到達可能なレジスタを演算器 j の入力レジスタ，出力から出力方向に到達可能なレジスタを演算器 j の出力レジスタとする．また, RTL 回路のテスト実行時に演算器 j に入力するテストパターンを印可するレジスタを演算器 j の入力テストレジスタ，その出力応答を観測するレジスタを演算器 j の出力テストレジスタと定義する．

図.1 に RTL データパスの部分回路を示す. 図.1 (a)の演算器 A に着目すると, 演算器 A の入力から他のレジスタを介さずに到達可能なレジスタは R0，R1，R2 である．したがって，R0，R1，R2 は演算器 A の入力レジスタである．また，出力側も同様に着目すると，R2，R3 は演算器 A の出力レジスタである．次に，図.1(b)に演算器 A をテスト生成する際に R0，R2 から入力を与え，R3 で出力応答を観測する例を示す．図.1(b)において，入力を与えた R0，R2 は演算器 A の入力テストレジスタ，出力応答を観測した R3 は演算器 A の出力テストレジスタである．

2.3 演算器のテスト集合

本論文では，RTL データパスに記述されている演算器 j 単体を対象としてテスト生成を実行したときのテスト集合を，演算器 j のテスト集合と定義する．このときのテストパターン数を，演算器 j の各入力および出力のテスト集合として定義する．

2.4 テストレジスタの衝突

RTL 回路中の演算器 A と B の入力が同じ入力テストレジスタに割当てられていた場合，一方の演算器のテスト集合で他の演算器のテストを保証できない．そのため，複数の演算器を同時にテスト実行するためには，それぞれの演算器のテスト集合を別々に入力する必要がある．また，出力レジスタも同様である．これをテストレジスタの衝突と定義する．入力レジスタのテストレジスタの衝突を入力テストレジスタの衝突，出力レジスタのテストレジスタの衝突を出力テストレジスタの衝突とする．

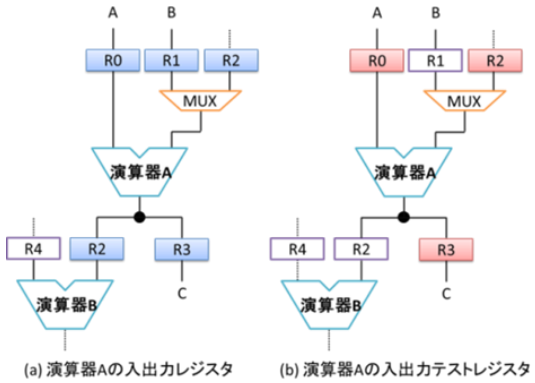


図.1 RTL データパスの部分回路

3. 演算器のテスト並列化によるテストパターン数削減

3.1 テストパターン数見積もり

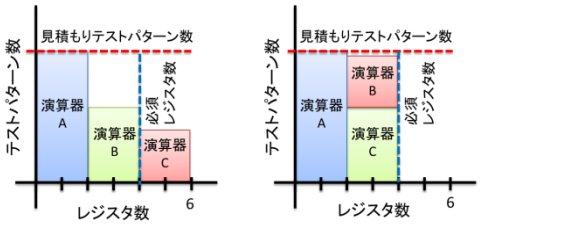
回路中の全演算器に対して同時に並列テストをおこなう場合，全演算器が同時にテストされるために演算器の入出力に対してテストレジスタを 1 対 1 で割当てる必要がある．この場合のテストパターン数はテストテストパターン数が最大の演算器のテストパターン数となる．しかしながら, 回路構造によっては RTL 回路中のレジスタ数が少ないために，すべての演算器の同時テスト並列化が実現可能とは限らない．そのため，テストレジスタの衝突が起きるようなテストレジスタ割当てやテストポイント挿入を用いたテストレジスタ割当てをおこなう必要がある. この場合の，最小のテストパターン数は割当てられたテストパターン数が最大のレジスタのテストパターン数となる．

しかしながら，テストレジスタの衝突を繰返すことによりテストパターンが増大してしまう問題が存在する．そのため，事前に最小のテストパターン数を見積もりテストレジスタ割当て時の制約とする必要がある．

3.2 節に演算器のテスト並列化を実現するためのレジスタ割当てについて述べる．3.3 節に 3.2 節の問題を解決するためのテストポイント挿入について述べる．

3.2 演算器のテスト並列化を実現するための演算器のテストレジスタ割当て

演算器の並列テスト時，演算器の入出力に対してテストレジスタを 1 対 1 で割当てが不可能な場合がある．その場合，見積もりパターン数を超えないようにテストレジスタの衝突を発生させることによって演算器のテスト並列化を実現する．このとき，1 つのレジスタに演算器 j の複数の入力を入力テストレジスタとして割当てないように注意する．



(a)レジスタ数6(=必須レジスタ数)のため，入力レジスタの衝突なし　(b)レジスタ数4(<必須レジスタ数)のため，入力レジスタの衝突考慮

図.2 入力テストレジスタの衝突による テストパターン数の変化

図.2 に入力テストレジスタ衝突によるレジスタ数の変化を示す．図.2 での演算器の並列テスト実行に必要なパターン数の見積もりは演算器 A のテストパターン数と同等となる．図.2(a)は RTL 回路中内のレジスタ数である必須テストレジスタ数を下回るため演算器のテスト並列化が不可能となる．一方で，図.2(b)は演算器 B と C の入力テストレジスタの衝突を発生された例である．この場合，見積もりパターン数を増大させることなく必須テストレジスタ数以内にレジスタ数を減少させることで．演算器の並列テストが可能となる．しかしながら，テストレジスタ割当ては演算器とレジスタ間に接続関係が存在している場合のみ可能であるという問題が存在する．

3.3 テストポイント挿入を用いた演算器のテストレジスタ割当て

演算器の並列テストを行う場合，並列テストのためのテストレジスタ割当てをおこなう必要がある．しかしながら，RTL 回路構造上の演算器の入出力とレジスタ間に接続関係が存在しない場合，並列テストのためのテストレジスタ割当てがなされない．そのため，テスト並列化がおこなえずテストパターン数の削減が見込めないことがある．この問題に対して，テストポイント挿入を用いたテストレジスタ割当てをおこなうことで問題を解決する．テストポイント挿入は接続関係のない演算器の入出力とレジスタ間に対してテストポイント(2 入力 MUX)を挿入することで既存の接続関係を維持したまま接続関係を作成する．しかしながら，テストポイントは面積オーバーヘッドも大きく，対象故障数の増加からテストパターン数が提案手法の削減効果以上に増大する可能性がある．したがって，演算器のテスト並列化を実現するためのテストポイント挿入数は最小化する必要がある．

3.4 問題の定式化

見積もりテストパターンの定式化とレジスタ割当ての定式化をおこなう．また，本手法で扱うデータパスの信号線のビット幅はすべて一様とし，すべての演算器の出力数は 1 とする．

- **見積もりテストパターン**

演算器に対して，並列テストを実行する時のテストパターン数を見積もる問題の定式化をおこなう．RTL 回路中には，演算器$j(j = 1, 2, \dots, M)$とレジスタ$l(l = 1, 2, \dots, N)$が存在し，その演算器jの入力数をI_j，テストパターン数をw_jとすると，その RTL 回路に対する並列テスト時テストパターン数の見積もりは改変した二次元ビンパッキング問題に帰着することが可能である．

[問題定式化]：見積もりテストパターン数

入力：幅I_j，高さw_jの長方形アイテムjを M 個$(j = 1, 2, \dots, M)$，アイテムを格納する幅N，高さ無限大のビン

出力：すべてのアイテムを格納した時の最大の高さ

制約 1：他のアイテムと重ならずにビンの幅を超えないようにアイテムを格納

最適化：Minimum（すべてのアイテムをビンに格納したときの最大の高さ）

- **テストレジスタ割当て**

➢ **定義 1：演算器テスト**

I_j入力 1 出力演算器jのテストパターンを演算器jテストとし，テストパターン数をw_jとする．演算器jのk番目の入力に設定するテストパターン集合を演算器j入力kテストT_{jk}と定義する．演算器jの出力で観測するテスト応答集合を演算器j出力テストT_jと定義する．

➢ **定義 2：演算器テストのスケジューリング**

演算器jの入力kテストと演算器j出力テストを幅 1，高さ 1 の正方形を，w_j個用いて表現する．演算器テストスケジューリンググラフは，行がテスト実行時刻を示し，列が入力テストレジスタと出力テストレジスタを示すグラフである．演算器jの入力kテストのu番目のテストパターンを入力テストレジスタℓのテスト実行時刻tに配置することを，演算器jの入力kテストuを入力レジスタℓの時刻tにスケジューリングすると定義する．

➢ **定義 3：諸定義**

演算器$j(j = 1, 2, \dots, M)$：Mは演算器数

演算器jの入力 $k(k = 1, 2, \dots, I_j)$：I_jは演算器 j の入力数

レジスタ$\ell(\ell = 1, 2, \dots, N)$：$N$はレジスタ数

テストパターン$u(u = 1, 2, \dots, w_j)$：w_jは演算器jの入力kテストのテストパターン数

I_{jk}：演算器jの入力 k 入力レジスタ集合

O_j：演算器jの出力レジスタ集合

T_{jk}：演算器j入力 k テスト

T_j：演算器j出力テスト

$S_{jku\ell}$：演算器jの入力kテストuを入力レジスタℓにスケジューリングした時刻

$S_{ju\ell}$：演算器j出力テストuをレジスタℓの出力テストレジスタにスケジューリングした時刻

$X_{jk\ell} \in \{0,1\}$：演算器j入力kテスト中の少なくも一つのテストパターンをレジスタℓの入力テストレジスタにスケジュールしたとき，レジスタ$\ell \notin I_{jk}$であれば1，それ以外の場合は0となる．

$Y_{j\ell} \in \{0,1\}$：演算器j出力テスト中の少なくも一つのテストパターンをレジスタℓの出力テストレジスタにスケジュー

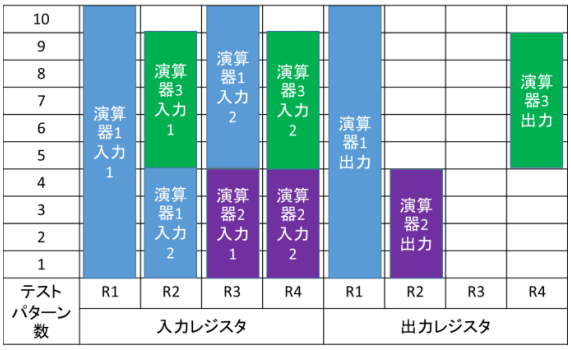


図.3 演算器テストスケジューリンググラフ

ルしたとき，レジスタ$\ell \notin O_j$であれば 1，それ以外の場合は 0 となる．

$g_{jku\ell t} \in \{0,1\}$：演算器$j$の入力$k$テスト$u$を入力レジスタ$\ell$にスケジューリングした時刻が時刻$t$である（つまり$S_{jku\ell} = t$）場合 1，それ以外の場合は 0 となる．

$h_{j\ell ut} \in \{0,1\}$：演算器j出力テストuをレジスタℓの出力テストレジスタにスケジューリングした時刻が時刻tである（つまり$S_{ju\ell} = t$）場合 1，それ以外の場合は 0 となる．

E_T: 見積もりテストパターン数

[問題定式化]：テストレジスタ割当て

入力：RTL 回路中のすべての演算器j入力kテスト T_{jk}，演算器j出力テストT_j，N個レジスタに対して，テストパターン数の上限を見積もりテストパターン数E_Tとした演算器のテストスケジューリンググラフ，演算器jの入力kの入力レジスタ集合I_{jk}，演算器jの出力レジスタ集合O_j

出力：演算器j入力kテスト T_{jk}と演算器j出力テストT_jがすべてスケジューリングされた演算器テストスケジューリンググラフ

制約 1：$\forall j, \forall k, \forall \ell, \forall u,\ S_{ju\ell} = S_{jku\ell}$

制約 2：$\forall j, \forall k, \forall \ell, \forall u, 1 \le S_{jku\ell} \le E_T$

制約 3：$\forall \ell,\ \forall t,\ \sum_j^M \sum_k^{I_j} \sum_u^{w_j} g_{jk\ell ut} \le 1$

制約 4：$\forall \ell,\ \forall t, \sum_j^M \sum_u^{w_j} h_{j\ell ut} \le 1$

$$\textbf{最適化}：\text{Minimum}\left(\sum_{j=1}^{M}\sum_{\ell=1}^{N}\left\{\left(Y_{j\ell}\right) + \left(\sum_k^{I_j} X_{jk\ell}\right)\right\}\right)$$

図. 3 に演算器テストスケジューリンググラフの例を示す. 左部の数字はテストパターン数を示し,「R1」から「R4」はレジスタ名を示し，「入力レジスタ」「出力レジスタ」は入力レジスタと出力レジスタを示す．グラフ中の矩形は各演算器入出力のテストパターンを示す．また，見積もりテストパターン数は 10 である．

3.5 コントローラ拡大

コントローラは演算器のテスト並列化を実現する RTL データパスへの制御信号を出力するとは限らない．したがって，演算器のテスト並列化を実現するためにコントローラ拡大をおこなう必要がある．コントローラ拡大とは，回路のテスタビリティを向上させるために状態や状態遷移を追加する手法である．

コントローラには，機能動作時には絶対に遷移し得ない無効状態が存在する場合がある．本論文ではスキャンテストを前提とするため，コントローラ中のスキャン FF がテスト時には無効状態にも遷移が可能となる．したがって，存在する無効状態を用いてコントローラ拡大を行うことで，テスト時にのみ遷移が可能な状態に，任意の制御信号を定義することが可能である．このとき，コントローラ拡大をおこなう無効状態をテスト無効状態と呼ぶ．なお，無効状態が不足する場合はスキャン FF を追加して無効状態を生成する．

演算器のテスト並列化を実現する制御信号は，演算器とその入出力レジスタの間に存在する MUX，割当てられたテストレジスタの情報を用いて，RTL データパスから容易に求めることが可能である．テストポイントが挿入された場合は，テスト無効状態でのみテストポイントが動作するように定義する．また，制御信号の割当てが異なる複数のテスト無効状態が存在することをテスト無効状態の衝突と定義する．テスト無効状態の衝突を削減することでコントローラ拡大に必要な無効状態数が削減される．

複数の入力テストレジスタが割当てられた演算器が存在する場合または，出力テストレジスタ衝突が存在する場合は複数のテスト無効状態を用いる．

3.6 演算器のテストレジスタ割当てアルゴリズム

3.4 の問題は, テストポイント挿入箇所とテストレジスタ割当て情報の最適な組合せが数多く存在する．また，探索空間は

$$\left(\prod_{j=1}^{M}\left(\prod_{k=1}^{I_j}(N)\,N\right)\right)^{M} E_T$$

となる．したがって，テストポイント挿入数を最小とするようなテストレジスタ割当てのヒューリスティックアルゴリズムを提案する．図.4 にテストレジスタ割当ての全体アルゴリズムを示す．本手法は RTL データパス D，テストパターン数 W，見積もりテストパターン数 ET が入力として与えられる．

はじめに，RTL データパス D とテストパターン数 W から，入出力レジスタ集合 I/O REG を求める(Line1)．表.1 に入出力レジスタ集合の例を示す．「A0」,「A1」,「M0」はそれぞれ演算器名を示し,「左入力」「右入力」「出力」はそれぞれの演算器の左入力，右入力，出力を示す．「R0」から「R3」は RTL データパス回路中のレジスタを示し,「○」は左部の演算器の入出力から上部のレジスタが他のレジスタを介さずに到達可能であることを示す．「TV 数」は 2.3 節で定義したテストパターン数である．次に，求めた入出力レジスタ集合 I/O REG と見積もりテストパターン数E_Tを用いて，演算器のテストパターン数の降順にテストポイントを用いないようなレジスタ割当てをおこないテスト無効状態情報 STATE INFO を求める(Line6)．テスト無効状態情報法はテスト無効状態の集合で構成される．また，テストレジスタ割当ての選択肢が存在した場合はランダムに割当てる．テストパターン数が大きい演算器からテストレジスタを割当てるためテストパターン数が少ない演算器は

```
1.  Additional state derivation(D, W, ET,)
2.  D:Data path, W:Weight,
3.  ET:Estimated Number of Test patterns
4.  {
5.      I/O REG = I/O REG Analysis(D, W);
6.      STATE INFO = Weight Priority Allocation
                (I/O REG, ET);
7.      STATE INFO = Allocation Exchange(STATE INFO, I/O REG);
8.      STATE INFO = State Marge(STATE INFO, I/O REG);
9.      return(STATE INFO);
10. }
```

図.4 全体アルゴリズム

表.1 入出力レジスタ集合

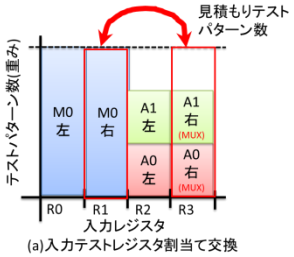
演算器名	入出力	R0	R1	R2	R3	TV数
A0	左入力			○		15
	右入力		○			
	出力			○		
A1	左入力	○		○		15
	右入力		○			
	出力			○	○	
M0	左入力	○				50
	右入力		○			
	出力		○			

表.2 テスト無効状態

演算器名	入出力	R0	R1	R2	R3	TV数
A0	左入力			○		15
	右入力		○			
	出力			○		
A1	左入力	×		○		15
	右入力		○			
	出力			×	○	
M0	左入力	○				50
	右入力		×		M	
	出力		○			

テストポイントも用いる可能性が高くなる．また，テスト無効状態を求めた際に見積もりテストパターン数よりテストパターン数が多くなる可能性が存在する．その場合，見積もりテストパターン数を求めた際の割当てをもとにテスト無効状態を求める．表.2 にテスト無効状態の例を示す．表.1 と同様「A0」，「A1」，「M0」はそれぞれ演算器名を示し，「左入力」「右入力」「出力」はそれぞれの演算器の左入力，右入力，出力を示す．「R0」から「R3」は RTL データパス回路中のレジスタを示し，「○」はテストレジスタであり，「×」はテストレジスタに選択されなかったレジスタである．「M」は演算器の入出力とレジスタに接続関係が存在しない場合にテストポイントの挿入によって演算器の入出力とレジスタの接続関係を作成しテストレジスタ割当てたことを示す．「TV 数」は 2.3 節で定義したテストパターン数である．また，テストパターン数 W は入力テストレジスタの衝突を考慮する場合の制約としても使用する．次に，テスト無効状態情報 STATE INFO，入出力レジスタ集合 I/O REG を使用し，演算器のテストパターン数の降順に演算器を対象にテストレジスタ割当て交換を行う(Line7)．最後に，テスト無効状態情報 STATE INFO，入出力レジスタ集合 I/O REG を用いて，テスト無効状態のマージをおこなう(Line8)．テスト無効状態情報 STATE INFO を出力してアルゴリズムを終了する．

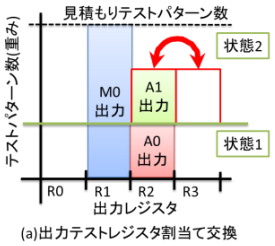
図.5 にテストレジスタ割当て交換の例を示す．例は入力テストレジスタ割当てのみで説明する．実際のテストレジスタ割当ては出力レジスタに対しても入力テストレジスタ割当てと同様の処理をする．表.1 の入力レジスタ集合に対してテストパターン数優先テストレジスタ割当てをおこなうと図.5 の(a)のようなテストレジスタ割当てとなる．テストポイント削減のために演算器のテストパターン数の降順



(b)テスト無効状態表(入力のみ)

演算器名	入出力	R0	R1	R2	R3	TV数
A0	左入力			○		15
	右入力		○	←	M	
A1	左入力	×		○		15
	右入力		○	←	M	
M0	左入力	○				50
	右入力		○	→	M	

図.5 レジスタ割当て交換の例



(c)状態2テスト無効状態(出力のみ)

演算器名	入出力	R0	R1	R2	R3	TV数
A0	出力			×		15
A1	出力			○	○	15
M0	出力		○			50

(b)状態1テスト無効状態(出力のみ)

演算器名	入出力	R0	R1	R2	R3	TV数
A0	出力			○		15
A1	出力			×	×	15
M0	出力		○			50

図.6 出力レジスタ割当て交換の例

にテストレジスタ割当ての交換をおこなうか解析する．テストパターン数が最大の演算器 M0 の右入力のレジスタ割当てとレジスタ R3 のテストレジスタ割当てに着目する．この 2 つのテストレジスタ割当てを交換することによりテストポイント数を 2 つから 1 つに削減可能である．図.5 の(b)に力テストレジスタ割当ての交換の例を示す．テストレジスタ割当ての交換により使用しなくなったレジスタ R3 と演算器 A0 の右入力の接続関係を作成するテストポイントとレジスタ R3 と演算器 A1 の右入力の接続関係を作成するテストポイントを削除し，新たに演算器 M0 の右入力にレジスタ R3 との接続関係を作成するテストポイントを挿入することでテストポイント挿入数を 1 つ削減する．次に，テストパターン数の少ない演算器を選択し同様の処理を繰返す．

図.6 にテスト無効状態のマージの例を示す．図.6 の(a)に出力レジスタ割当ての例を示す．状態 1 と状態 2 においてレジスタ R2 は 2 つの演算器の出力テストレジスタの衝突がおきているためテスト無効状態の衝突が発生するためマージは不可能である．そのため，テストレジスタ割当ての交換をおこなうことでテスト無効状態の衝突を解消し状態のマージを試みる．演算器 A1 の出力レジスタに着目する．演算器 A1 はレジスタ R3 に対しても接続関係を持っているためレジスタ R2 からレジスタ R4 への出力テストレジスタの割当ての交換が可能である．図.6 の(b)と(c)に出力レジスタ割当ての交換を示す．レジスタ割当ての交換によって出力テストレジスタの衝突が解消されたことにより，テスト無効状態の衝突を解消され状態 1 と状態 2 のテスト無効状態のマージが可能となる．また、1 つの入力に複数の入力テストレジスタが割当てられた演算器が存在する場合も同

様の方法でテストレジスタ割当ての交換を試みる.

以上が,テストレジスタ割当てアルゴリズムの例である.状態のマージをおこなうと表.2 のようなテスト無効状態となる.

4. 実験結果

本章では，フルスキャン設計が施された 13 個の RTL 回路に対して，通常回路，先行研究手法適用回路，提案手法適用回路を作成し各回路に対して実験を行い，テストパターン数，面積オーバーヘッドに対して評価した．RTL 回路生成のための動作合成ツールは動作合成には内製の動作合成ツール PICTHY を使用し，信号線のビット幅は 32 ビットとした. 論理合成ツールは Synopsys 社の Design Compiler を使用し，ATPG は同じく Synopsys 社の TetraMAX を使用し，対象故障モデルは単一縮退故障とした．テスト生成のバックトラック数は 10,000,000 回に設定した ．また，故障検出効率 100.00%に到達しない場合，未検出故障に対してテスト生成のバックトラック数を 1,000,000,000 回に設定し再度テスト生成をおこなった.

表.3 に実験結果を示す.「回路名」は実験対象の RTL 回路名を示し,「without」は通常回路に対する実験結果を示し,「[8]」は先行研究手法適用回路での実験結果を示し,「propose」は本手法適用回路の実験結果を示す.「farget faults」は対象故障数を示し,「detect」は検出故障数を示し,「abort」は打切り故障数を示し,「area」は回路面積を示し,「ATPGtime」はテスト生成時間を示す.「FC」と「FE」はそれぞれ故障検出率と故障検出効率を示し,「ETV」は演算器を並列にテストした際のテストパターン数の見積もりを示す.「TP 挿入数」は制御点と観測点の挿入数を示し,「追加状態数」はコントローラ拡大時に追加した状態数を示す.「TV 数」はテストパターン数を示す。「*」はテスト生成時のバックトラック回数を 100,000 回に設定しテスト生成をおこなった．また,「**」は未検出故障に対する 2 度目のテスト生成テスト生成をおこなっていない.

本論文の目的であるテストパターン数は通常の回路に対して平均で約 20%，最大で約 82%削減することができた．この値は通常の回路のテストパターン数と提案手法の打切り故障とテストパターン数の和を比較したものである．回路規模が大きくなるほど高い効果が得られる傾向が確認できた．また，先行研究のテストパターン数削減率と同等の結果を得られた．回路面積オーバーヘッドは通常の回路に対して平均約 6.5%となり, 先行研究の回路面積オーバーヘッドに対して平均約 6.5%，最大約 31%削減できた.

5. おわりに

本論文では，階層を維持しない並列テストのための DFT 手法を提案した．評価実験では 13 個中 12 個の回路のテストパターン数の削減に成功し，8 個中 7 個の回路の面積オーバーヘッドの削減に成功した.

謝辞 謝辞本研究は一部，日本学術振興会科学技術研究補助金基盤(C)(課題番号 26330071,15K06086)の研究助成による.

参考文献

[1] S. Kajihara, I. Pomeranz, K. Kinoshita :"Cost-Effective Generation of Minimal Test Sets for Stuck-at Faults in Combinational Logic Circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems , Vol.14, Issue12, pp.1496-1504, Dec.1995.

[2] S Kajihara,I. Pomeranz,K. Kinoshita and S. M.Reddy "On Compaction Test Sets by Addition and Removal of Test Vectors," VLSI Test Symposium, 1994. Proceedings., 12th IEEE, pp.202-207,Cherry Hill,NJ, The USA, Apr 1994.

[3] M. J. Geuzebroek, J. Th. van der Linden, and A. J. van de Goor, "Test Point Insertion for Compact Test Sets," Test Conference, 2000. Proceedings. International, pp.292-301, Atlantic City NJ, The USA, Oct 2000.

[4] Santiago Remersaro, Janusz Rajski, Thomas Rinderknecht, Sudhakar M. Reddy, Irith Pomeranz, "ATPG Heuristics Dependant Observation Point Insertion for Enhanced Compaction and Data Volume Reduction," 2008 IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems, pp.385-393, Boston MA , The USA, Oct.2008.

[5] 吉村正義, 細川利典, 太田光保, "ATPG パターン数削減指向テストポイント挿入方法," 電子情報通信学会論文誌, Vol. J86-D-I, pp. 884-896, Dec.2003.

[6] Kedarnath J. Balakrishnam and Lei Fang, "RTL Test Point Insertion to Reduce Delay Test Volume," 25th IEEE VLSI Test Symposium (VTS'07), Berkeley CA , pp.325-332, The USA , May.2007.

[7] B.T. Murray and J.H. Hayes, "Hierarchical test gen-eration using pre computed tests for modules," IEEETrans. Computer-Aided Design of Integrated Circuits and Systems, vol.16, no.9, pp.1001–1014, 1990.

[8] 大崎直也,細川利典,山崎紘史,吉村正義,"テストパターン数削減のための RTL テストポイント挿入法," ディペンダブルコンピューティング研究会, pp.43-48, Dec.2016.

[9] H. Fujiwara, Logic Testing and Design for Testability, MIT Press Cambridge, MA, The USA , Sept.1985.

[10] M. Abramovici, M. A. Breuer, and A. D Friedman, Degital Systems Testing and Teaable Design, Computer Science Press, Sept.1990.

表.3 実験結果

回路名	without								[8]												Propose									
	target faults	detect	abort	area	ATPG time(s)	FC(%)	FE(%)	TV数	target faults	detect	abort	area	ATPG time(s)	FC(%)	FE(%)	TPI挿入数 制御	TPI挿入数 観測	reg数	TV数	ETV数	target faults	detect	abort	area	ATPG time(s)	FC(%)	FE(%)	MUX 挿入数	追加 状態数	TV数	ETV数
ARF	42501	41477	0	22910	78531.50	97.59	100.00	661												67	57504	57504	0	31487	5315.58	100.00	100.00	0	1	120	67
BPF	28114	28109	4	15379	757420.65	99.99	99.99	446												67	34041	34041	0	19026	84930.48	100.00	100.00	0	1	146	67
ex4	16427	16427	0	9491	965.91	100.00	100.00	85	17013	17013	0	9954	2.06	100.00	100.00	1	0	1	102	67	16685	16685	0	9602	1494.87	100.00	100.00	1	1	84	67
ex2	17087	17085	3	9571	504274.95	99.99	99.99	87	16939	16939	0	9597	29.31	100.00	100.00	0	0	0	84	67	17100	17098	2	9597	4968.53	100.00	100.00	0	2	82	67
dct	6687	6687	0	4382	0.07	100.00	100.00	33	9183	9183	0	5940	0.11	100.00	100.00	6	2	6	32	28	7128	7128	0	4531	0.04	100.00	100.00	2	2	33	28
dfct	38084	38084	0	20929	800.5	100.00	100.00	112	39921	39920	0	22145	1488.67	99.99	100.00	2	2	2	95	67	40158	40158	0	21989	3991.17	100.00	100.00	1	2	96	67
DWT_MPEG	57102	57101	0	33573	20542.7	100.00	100.00	163												67	57291	57291	0	33487	22358.8	100.00	100.00	0	1	144	67
FIR_MPEG	61401	58401	2311	36212	104223.45	95.11	96.19	207*												67	61660	61652	1	36312	546021	99.99	99.99	0	1	182	67
FFT	42065	42065	1	23334	1961.83	99.99	99.99	182	42410	42408	2	24288	356652.97	99.99	99.99	0	2	2	138	67	42405	42402	0	23609	7161.33	99.99	100.00	0	2	134	67
kim	19462	19462	0	6556	19.12	100.00	100.00	124	10734	10734	0	6900	22.83	100.00	100.00	0	1	1	120	36	10585	10587	2	8595	697.95	99.98	99.98	0	2	101	36
maha	7685	7685	0	5049	0.14	100.00	100.00	188	7961	7928	0	5406	0.39	99.59	100.00	1	0	1	188	66	7711	7710	0	5059	4.26	99.99	100.00	1	0	185	66
sehwa	8856	8856	0	5878	0.15	100.00	100.00	221												66	8882	8881	0	5879	1002.39	99.99	100.00	0	0	220	66
fig17	59615	59483	4	31074	78393.19	99.99	99.78	637**				35350				4	3	4		161	65413	65283	84	34526	412762.44	99.87	99.8	2	2	430**	161