論理回路に対するテスト実行時間削減法*

Reduction of test application time for logic circuits

樋上喜信†　梶原誠司‡　市原英行§　高松雄三†

Yoshinobu HIGAMI† , Seiji KAJIHARA‡ Hideyuki ICHIHARA§ and Yuzo TAKAMATSU†

Recently, reduction of test application time is one of the most important challenges in the VLSIs testing field. This is because long test application time increases the test costs. In this article, we survey recent researches for reducing test application time, which include test compaction for combinational circuits and non-scan sequential circuits and test application time reduction for scan circuits.

Key words: VLSI, Test application time reduction, Test compaction

1. まえがき

論理回路のテストコストには，テスト実行時間，テスト生成やテスト容易化のためのコスト，テスタのコストなど様々な要因がある．近年，テストデータ量の増大によるテストコストの増大が深刻な問題となっている．テストベクトル数やテストデータ量を削減することは，テスト実行時間，テスタのメモリ容量制約などの観点から重要な課題である．特に最近の 10 年程の間にその技術は急速に進歩し，現在でもなお LSI テスト分野における主要な研究トピックスの一つである．

LSI テスタに格納するテストデータ量の削減や，テストベクトル数を削減する手法は，テスト圧縮法とよばれている．テスト圧縮法にはテストコンパクション (test compaction) とテストコンプレッション (test compression) の二通りのアプローチがあり，それぞれに優れた手法が提案されている§ ．

テストコンパクションは，テスト生成において用いられる手法であり，高い故障検出率のテストパターン¶ をできるだけ少ないテストベクトル数で実現することを目標とする．テスト実行時間は，一般的にはテストベクトル数に比例するため，テストコンパクションによりテストベクトル数を削減することによって，同時にテスト実行時間を削減することができる．

本稿では，テストコンパクションに基づくテスト実行時間削減法について概説する．本稿の構成は以下のとおりである．まず，縮退故障を対象にしたテストコンパクションとして，２．では組合せ回路に対するテストコンパクションについて，３．ではスキャン設計を施さない順序回路に対するテストコンパクションについて述べる．４．では，スキャン回路に対するテスト実行時間削減法について述べる．最後に５．で本稿のまとめと今後の課題について述べる．

---

* 電子情報通信学会和文誌 D-I, Vol. J87-D-I. No.3, pp. 291-307, 2004. の第 1, 2, 3, 6, 7 章より引用した.

† 愛媛大学工学部 情報工学科

† Department of Computer Science, Faculty of Engineering, Ehime University

‡ 九州工業大学情報工学部電子情報工学科

‡ Department of Computer Science and Electronics, Kyushu Institute of Technology

§ 広島市立大学情報科学部情報機械システム工学科

§ Faculty of Information Sciences, Hiroshima City University

§ 英語では２つの用語は区別されてきたが，日本語ではどちらもテスト圧縮とされている

¶ 本論文では，テストベクトルの集合をテストパターンとよぶ.

2. 組合せ回路のテストコンパクション

テストコンパクション技術は，一旦生成したテストパターンを小さくするための手法と，最初から少ないテストベクトル数になるようにテストパターンを生成するための手法に大別される．本節では，組合せ回路のテストベクトル数削減について代表的な手法を説明する．

2.1 生成されたテストベクトル数の削減

テストコンパクションは，高い故障検出率のテストパターンをできるだけ少ないテストベクトル数で実現する技術である．故障検出率を保障しつつテストベクトル数を少なくするには，各テストベクトルができるだけ多くの故障を検出すること，および，テストパターン中に無駄なテストベクトルを含まないことが必要である．一つのテストベクトルが多くの故障を検出できるように，生成した複数のテストベクトルを一つに統合する手法として，静的圧縮[1)]がある¶．通常，ある故障に対して生成したテストベクトルには，値が未設定の入力がある．静的圧縮では，この未設定値を利用して，テストベクトルを統合する．たとえば，ある故障に対して生成されたテストベクトルが 0x10，他の別の故障に対して生成されたテストベクトルが x1x0 とすると，これら 2 つのテストベクトルは 0110 の一つのテストベクトルに統合可能である．静的圧縮を適用するには，テストベクトルの生成後に未設定入力値を残しておかなければならない．

生成したテストパターンには意図することなく無駄なテストベクトルが含まれることがある．たとえば，ある回路に対するテストパターンが，表 1(a) のように $t_1, t_2, ..., t_5$ の順に生成されたとする．テスト生成では，それまでに生成したテストパターンでは検出できていない未検出故障を対象にテストベクトルを生成するため，各テストベクトルはそれを生成した時点では無駄ではない．しかしながら，対象とした故障がそれ以降に生成されるテストベクトルで検出されるなら，そのテストベクトルは無駄となる．表 1(a) の例では，t_1 で検出される故障が他のテストベクトル t_2 と t_5 で検出されるため，t_1 は不必要である．このようなテストベクトルは，冗長テストベクトルと呼ばれる[2)]．冗長テストベクトルをテストパターンから除去する手法として，逆順故障シミュレーション[3)]が知られている．これは，いったん生成したテストパターンを，生成したテストベクトルの順序と逆の順序で故障シミュレーションを実施することで，冗長なテストベクトルを識別・削除する手法である．表 1(b) に示すようにテストベクトルを逆順にしてシミュレートすれば，t_1 は新たに未検出故障を検出しないため，冗長と判断でき除去できる．しかしながら，逆順故障シミュレーションはすべての冗長テストベクトルを除去できるわけではない．たとえば，表 1(b) においても t_3 は冗長であるが，逆順故障シミュレーションでは除去できない．テストパターン中のすべての冗長テストベクトルを少ない計算手間で識別・除去する方法として二重検出法[2, 4)]が提案されており，その効率的なプログラム実装についても研究が進んでいる[5, 6)]．なお，逆順故障シミュレーションは，縮退故障のテストのように，故障を検出するテストベクトルが前後のテストベクトルとは無関係のときに適用可能である．しかし，本節で述べる他のテスト圧縮手法の考え方は，二重検出法も含め何らかの故障モデルを前提としたテスト生成であればよく[7)]，故障モデルからは独立している．

テストパターンが冗長なテストベクトルを一つも含んでいない（すなわち，そのテストパターンからど

¶ 静的圧縮という用語は広義には，一旦生成されたテストベクトル数を削減する技術の総称として用いられることもある．本節では，文献[1)]で述べられた手法を狭義に静的圧縮とする．

表 1: テストパターンの例

(a) テスト生成順

テスト ベクトル	検出 故障
t_1	f_1, f_2
t_2	$(f_2), f_3$
t_3	$(f_3), f_4$
t_4	$(f_4), f_5$
t_5	$(f_1), f_6$

(b) 逆順

テスト ベクトル	検出 故障
t_5	f_6, f_1
t_4	f_4, f_5
t_3	$f_3, (f_4)$
t_2	$f_2, (f_3)$

(c) 極小テスト集合

テスト ベクトル	検出 故障
t_2	f_2, f_3
t_4	f_4, f_5
t_5	f_1, f_6

(d) 極小テスト集合の最小化

テスト ベクトル	検出 故障
t_2	f_2, f_3
t_6	f_1, f_4, f_5, f_6

のテストベクトルを削除しても故障検出率が低下する）とき，そのテストパターンを極小テストパターンという [2)]．たとえば，表 1(c) の $\{t_2, t_4, t_5\}$ で構成されるテストパターンは極小である．極小テストパターンであっても，一部のテストベクトルを他のテストベクトルに置き換えれば，さらにテストベクトル数を少なくできる [2, 4, 8)]．たとえば，故障 f_1, f_4, f_5, f_6 をすべて検出するようなテストベクトル t_6 を生成できれば，t_4, t_5 の代わりとなるため，表 1(d) の $\{t_2, t_6\}$ というさらに小さなテストパターンができる．テストベクトルの置き換えは，次節で述べる動的圧縮を利用したテスト生成を伴うため，非常に時間のかかる処理であるが，テストパターンを最小に近づける強力なテストコンパクション手法である．

2.2 小さな初期テストパターンの生成

一旦生成されたテストパターンを圧縮する場合，圧縮前のテストパターンの大きさとともに処理時間が増加する．テストパターンの極小化のみなら，故障シミュレーションが基本であるため処理はまだ高速であるが，圧縮前のテストパターンが大きいと，極小化しても極小値は大きくなる．こうした問題を避けるには，最初に生成する初期テストパターンをできるだけ小さくする必要がある．そのためにはテストベクトルが検出する故障数を多くし，さらに，検出される故障は他のテストベクトルが検出しない故障を多く検出することが望ましい．各テストベクトルで検出する故障数を増加させるための代表的な手法は動的圧縮‖ である [1)]．動的圧縮は，ある故障 f_1 に対して生成されたテストベクトルに残っている未設定の入力値を用いて，他の未検出故障 f_2 を検出できるようにテスト生成を繰り返す手法である．f_2 に対するテスト生成では，入力値として f_1 を検出するために割り当てた入力値は保持するという制約がある．そのようなテストベクトルが生成できない場合は，無駄な計算時間が大きくなる．しかしながら，異なる複数の故障を検出するテストベクトルの生成能力は静的圧縮よりも高く，圧縮の効果は大きい．

テストパターンを生成する過程で，異なるテストベクトルが同じ故障を検出しないようにする工夫としては，巡回後方追跡 (rotating backtrace)[9)] が知られている．PODEM[10)] の信号値正当化操作である後方追跡において，同じ経路ばかりを通らないように，テスト生成の途中にファンインリストの順番を変化させることで, 入力値の偏り（類似のテストベクトルの生成）を防ぎ，結果として，他のテストベク

‖ 動的圧縮という用語は広義には，テスト生成中にテストコンパクションを行う技術の総称として用いられることもある．本節では文献 [1)] で述べられた手法を狭義の動的圧縮とする．

トルと異なる故障を検出するようなテストベクトルの生成可能性を高めることができる．テスト生成アルゴリズム中に容易に実装できる上，巡回後方追跡の考え方は，故障伝搬経路の選択やDアルゴリズム[11]の一致操作にも適用できる．通常のテスト生成の正当化操作や故障伝搬操作では，信号線選択にテスト容易化尺度 (testability measure) を用いるが，それと比較すると巡回後方追跡を適用した場合，冗長故障判定は難しくなることが欠点となる．

そのほかに，冗長なテストベクトルの生成を抑えるため，独立故障集合[12]を用いてテスト生成の対象故障を選択する方法[2, 9]も知られている．この手法は，大規模回路に適用するには，独立故障集合を求める計算で必要となる記憶領域が大きくなる点が問題であり，実用的には部分的な適用[9]が考えられる．

また，少ないテストベクトル数で高い故障検出率を得るには，回路内にテストポイントを挿入することも有効である[13, 14, 15]．テストポイント挿入はテスト生成だけでなく，論理 BIST における故障検出率向上にも効果を発揮する[16]．テストポイントには，信号値の設定に自由度を与える制御点と，信号値情報を取得するための観測点の二種類がある．テストポイントは付加回路により実現され，回路の遅延にも影響するため，少ないテストポイント数で，その効果が大きく得られる箇所を探すことが課題となる．

3. 順序回路のテストコンパクション

スキャン設計を施さない順序回路に対して，テスト生成後にテストコンパクションを行う静的圧縮法と，テスト生成中にテストコンパクションを行う動的圧縮法について説明する¶．

3.1 静的圧縮法

順序回路に対するテスト系列においては，たとえ故障を検出しないテストベクトルであっても，回路を特定の状態に遷移させる役割のあるテストベクトルを削除した場合，故障検出率が低下する可能性がある．静的圧縮法においては，テストベクトルを印加する際の状態が元のテスト系列のものと変化しないこと，あるいは，状態が変化しても元と同じ故障検出が可能なことを，故障シミュレーション等により保証しなければならない．そこで以下の節では，テストコンパクションの処理の途中で故障シミュレーションを行わなくても元の故障検出率が保証される手法と，故障シミュレーションによって元の故障検出率を保証する手法に分類し説明する．

3.1.1 故障シミュレーションを用いない手法

テスト系列からテストベクトルの削除や並べ替えなどの処理を行う際に，故障シミュレーションを行わなくても元の故障検出率が保証される場合がある．そのような手法の 1 つは，テスト系列を部分テスト系列に分割し，その後，他の部分テスト系列の状態遷移に影響を与えないように部分テスト系列を変更することである．

例えば，テスト系列 $T = Ts_1 - Ts_2 - Ts_3$ が与えられたとする．Ts_1, Ts_2, Ts_3 は部分テスト系列であり，$-$ は連結を表す．また，Ts_1, Ts_2 を印加した後の正常回路の状態を s_1, s_2 とし，故障 f_1 が Ts_1 で，f_2 が Ts_3 で検出され，他の部分テスト系列では検出されないとする．また，f_1, f_2 以外で検出される故

¶ 本節では広義の静的圧縮法，および広義の動的圧縮法を述べる．

障はないとする．このとき，もし s_1 と s_2 が同一で，かつ故障 f_2 が Ts_2 で顕在化されないならば故障マスクの問題が起こらず，Ts_2 を削除することができる．このとき故障シミュレーションを行わなくても，故障 f_1, f_2 の検出は保証される．文献 [17, 18] では，各テストベクトルを印加後の状態と，各故障が顕在化および検出される時刻を求め，これらの情報を用いて，上記のような同一の状態に遷移し，かつ故障検出に影響を与えないような部分テスト系列を削除している．また上記の例で，状態 s_2 がリセット状態と同一の場合，Ts_2 をリセット信号で置換することによってテスト系列が短くなる [19]．

初期状態が未知の状態を仮定した複数の部分テスト系列が与えられた場合，印加順序の変更や部分テスト系列の削除は，他の部分テスト系列の故障検出に影響を与えないため，故障シミュレーションは不要である．文献 [20] では，2 つの部分テスト系列を比較し，未設定値を利用することで，一部または全部のテストベクトルをオーバーラップして 2 つの部分テスト系列をマージする．マージする際，どの部分テスト系列も未知の初期状態を仮定しているので，状態遷移を考慮する必要はない．ただし，元のテスト系列に多くの未設定値が含まれていない場合には，テスト系列長短縮の大きな効果は期待できない．また，テスト系列長の最小化に，遺伝的アルゴリズムを用いて部分テスト系列の印加順序を決定する手法 [21] や，無閉路順序回路に対して生成した部分テスト系列をテンプレート (0，1 に設定すべき外部入力線の情報) として表現し，それらを重ね合わせる静的圧縮法 [22] においても，故障シミュレーションは不要となる．

3.1.2 故障シミュレーションを用いる手法

与えられたテスト系列 T に対して，短いテスト系列 T' を求め，T' に対して故障シミュレーションを行い，もし，故障検出率が T と同じ，またはより高い場合，T の代わりに T' をテスト系列として採用する手法がある．この操作において，短いテスト系列 T' をいかに求めるかが処理の性能を決める鍵となる．例えば，テストベクトルを削除する手法 [23, 24]，テストベクトルを復帰させる手法 (vector restoration)[25, 26, 27] がある．テストベクトルを復帰させる手法とは，与えられたテスト系列から，未知の状態を初期化するテストベクトルを除いて全てのテストベクトルを削除した後，選択したテストベクトルをテスト系列に復帰させる手法である．復帰させるテストベクトルとしては，故障を検出する時刻のテストベクトルから順に，前の時刻にさかのぼって選択する．テストベクトルを復帰させる手法は，高い圧縮の効果があり，計算時間を短縮する手法を組み込むことによって，実用的な計算時間で非常に短いテスト系列を得ることができる．

元のテスト系列に何らかの変更を施し，故障シミュレーションを行うことで，テストコンパクションを実現する手法がある．例えば与えれたテスト系列を T，その長さを n，T から得られたテスト系列を T' とする．T' に対して故障シミュレーションを行った結果，$n'(n' < n)$ 番目のテストベクトルで，T と同じ故障検出率が得られたならば，$n'+1$ 番目以降のテストベクトルが不要となり，テスト系列が短縮する．テスト系列 T' を得るための手法として，文献 [28] では，元のテスト系列 T を T_1, T_2 の 2 つの系列に分割した後，それらを並べ替えることによって，$T' = T_2 - T_1$ を得る．分割の際には，系列 T_2 の長さが全体の数%となるように分割することで，計算時間の増大を防いでいる．また，元のテスト系列 T 中のテストベクトルを別の位置にコピー・挿入することによって T' を求める手法も提案されている [23]．ただし，コピーするテストベクトルの選択や最適な挿入位置を求めることが難しく，総当たり的に行った場合には，計算時間の増大が問題となる．

3.2 動的圧縮法

組合せ回路の場合と同様，順序回路においても，1 つの部分テスト系列で検出される故障数をできるだけ多くすることで，テスト系列全体を短くすることができる．そこで，テスト生成の途中で生じた未設定値の外部入力線に対して，できるだけ多くの未検出故障を検出するように 0 または 1 を割り当てる手法として，各種の最適化手法を応用した手法が提案されている．

文献 [29] では，未設定値の外部入力線に，ランダムに 0 または 1 を割り当てたテストベクトルを複数生成し，それらに対して故障シミュレーションを行い，検出故障数が最大となるテストベクトルを選択する．またテストベクトルの選択に，遺伝的アルゴリズムを用いることで，できるだけ多くの故障が検出されるようなテストベクトルを選択する手法もあり，ランダムに割り当てた候補の中から選択するより，高い効果をあげている [30]．

組合せ回路のテスト生成における動的圧縮と同様に，未検出故障の中から選択した故障を検出するように，テスト生成アルゴリズムを適用して未設定値の外部入力線に値を割り当てる手法もある．このとき，効率良く故障を選択することによって，テスト生成時間を短縮する手法が提案されている [31]．また，静的圧縮法で用いられる手法を用いた動的圧縮法が文献 [32] で提案されている．ここでは，テストベクトルの省略や挿入を行うことで，テスト生成の途中で部分テスト系列の短縮を実現している．

動的圧縮法を単独で用いた場合，静的圧縮法で得られるテスト系列ほど短いテスト系列が得られない場合が多く，実用的には，動的圧縮法で得られたテスト系列に対して静的圧縮法を適用して，より短いテスト系列を得ることが効果的であると考えられる．

4. スキャン回路に対するテスト実行時間削減

回路の大規模化に伴うフリップフロップ (FF) 数の増大やテストベクトル数の増大によって、スキャン回路に対するテスト実行時間が非常に増大している．スキャン回路におけるテスト実行時間は，スキャンチェーンが 1 本の場合，

$$TAT = \#Vec + \#FF \times (\#Vec + 1) \tag{1}$$

で与えられ，ここで，$\#Vec$ は組合せ回路部分に印加されるテストベクトル数を，$\#FF$ は FF 数を表す．この式の第 1 項は，組合せ回路部分へのテストベクトルの印加を表し，第 2 項は，FF の値を設定および観測するためのスキャンシフト動作に必要なテスト実行時間を表す．従って，テスト実行時間を短縮するためには，第 1 項のテストベクトル数を削減するか，または，第 2 項のスキャンシフト動作に必要な時間を短縮することが考えられる．

組合せ回路部分のテストベクトル数を削減する研究については，先に述べたので，以下では，スキャンシフト動作に着目してテスト実行時間を短縮する手法について，スキャンチェーンの本数が 1 本の場合 (単一スキャンチェーン) と，複数本の場合 (多重スキャンチェーン) に分類して説明する．

4.1 単一スキャンチェーン

スキャンシフト動作に必要なテスト実行時間を表す (1) 式第２項は，全てのテストベクトルに対して，全ての FF にデータをスキャンイン・スキャンアウトすることを意味している．従って，テスト実行時間を短縮するための手法として，全てのテストベクトルに対して，全ての FF をスキャンシフトするのではなく，一部の FF のみにデータをスキャンイン・スキャンアウトする，または，一部のテストベクトルに対して，スキャンイン・スキャンアウトを省略することによって，テスト実行時間の削減が実現できる‖．例えば、表 2 のような 2 つのテストベクトル t_1, t_2 が与えられたとする．ここで，PI_1, PI_2 は外部入力値を表し，FF_1, FF_2, FF_3 は FF に設定すべき値，X はドントケア値を表す．もし，テストベクトル t_1 を印加後に故障の影響がすべて外部出力に伝搬したならば，t_2 の印加前に FF_1 のみにデータをスキャンインするだけでよく，スキャンシフトクロック数が減少し，テスト実行時間を短縮できる．

表 2: テストベクトルの例

	PI_1	PI_2	FF_1	FF_2	FF_3
t_1	1	0	1	1	0
t_2	1	1	0	X	X

特別なハードウエア機構を用いて，一部の FF のみにデータをスキャンイン・スキャンアウトする手法が提案されている．文献 [33] では，FF を２つのグループに分割し，一部のテストベクトルに対しては，第 1 グループのみスキャンシフトを行い，その他のテストベクトルに対しては，全ての FF をスキャンシフトする．また，文献 [34] では，スキャンシフトを行う FF を 1 から n(FF 数) まで可変にするようなハードウエア機構を提案している．例えば，あるモードでは 1 つの FF のみをスキャンシフトし，また別のモードでは，2 つの FF をスキャンシフトするというように，スキャンシフトを行う FF を順に 1 つずつ増加させる** ．

上記の 2 つの手法とは異なり，FF の値の観測に着目してスキャンシフト動作を省略するようにハードウエア機構を付加する手法が提案されている [35]．ここでは，FF のパリティを観測するような回路を付加することでそれを実現している．

特別なハードウエア機構を用いないで，1 部の FF のみにデータを制御・観測する手法が提案されている [36, 37]．この手法では，各テストベクトル毎にスキャンシフトを行う FF 数を可変にしている．例えば，5 つの FF が $FF_1, FF_2, ..., FF_5$ の順にスキャン入力側から並んでいるようなスキャンチェーンに対して，スキャンシフトクロック数を 2 とした場合を考える．このとき，FF_1, FF_2 にはスキャン入力から任意の値が設定され，同時に，スキャン出力側に近い FF_4, FF_5 の値が観測される．FF_3, FF_4, FF_5 の値はそれぞれ，FF_1, FF_2, FF_3 にあった値がシフトされ設定される．文献 [37] では，与えられたテストパターンに対して，各 FF の制御の必要度と観測の必要度を計算し，制御の必要度の高い FF をスキャン入力側に，観測の必要度の高い FF をスキャン出力側に配置して，スキャンチェーンを構成している．またテストベクトルの印加順序についても，スキャンシフトクロック数が少なくなるように最適化している．

通常のスキャン回路に対するテストでは，各テストベクトルを印加する前後にスキャンシフトを行う

‖ ここで述べる手法はフルスキャン設計を仮定しており，パーシャルスキャン設計とは本質的に異なる．

** これらの手法では，ハードウエア的には複数のスキャンチェーンが存在するが，スキャン入力とスキャン出力はともに 1 本であり，テスト時の動作およびテスト生成においては，単一スキャンチェーンとして扱うことができる．

が，一部のテストベクトルに対して，スキャンシフトを行わないで印加することによってテスト実行時間を削減する手法について以下で説明する．

順序回路用テスト生成法を用いた手法として，フルスキャン回路に対する手法[38]とパーシャルスキャン回路に対する手法[39]が提案されている．n 個のスキャン FF がある場合，全ての FF を制御または観測するためには n クロック必要であるが，通常動作状態において n より少ないテストベクトルを印加することで，FF の値の設定や観測ができたならば，テスト実行時間を削減できたことになる．ここでは，各故障に対して，検出困難性の評価値を用いて，スキャンシフトを行うかどうかを判断している．検出容易なものに対してはスキャンシフトを行わず，検出困難なものに対してのみスキャンシフトを行っている．

組合せ回路用テスト生成法で生成されたテストベクトルに対して，スキャンシフトを省略して複数のテストベクトルを連結する手法が提案されている[40]．例えば，テストパターン $T = \{t_1, t_2, t_3, t_4\}$ が与えられたとする．各テストベクトル t_i に対して，s_in_i, s_out_i をそれぞれ，t_i 印加前にスキャンインすべき状態ベクトル，t_i 印加後にスキャンアウトされる状態ベクトルとする．もし，t_1, t_2 が連結するペアとして選択されたとすると，s_in_1 をスキャンインし，$t_1 - t_2$ を印加した後，s_out_2 をスキャンアウトする．その後さらに，部分テスト系列 $t_1 - t_2$ とテストベクトル t_3 や t_4 が，スキャンシフトを省略して連結可能か調べる．

文献[40]の手法と逆の発想で，スキャンシフトを行わないテスト系列に対して，スキャンシフト操作を追加する手法が提案されている[41]．この手法では，まずスキャンシフトを行わない，通常の順序回路としてのテスト系列（T_0 と表す）を生成する．次にテスト系列 T_0 に対して，最も多くの故障が検出されるような，スキャンインおよびスキャンアウトのタイミングを求める．また，不要なテストベクトルを削除することによって，さらにテスト実行時間を削減する．

4.2 多重スキャンチェーン

複数本のスキャンチェーンを並列に動作させる手法またはそのような回路機構を，多重スキャンチェーンまたは並列スキャンチェーンとよぶ．多重スキャンチェーンの回路に対するテスト実行時間は，

$$TAT = \#Vec + max\_FF \times (\#Vec + 1) \tag{2}$$

となる．ここで，$\#Vec$ は組合せ回路部分に印加されるテストベクトル数を，max_FF は，スキャンチェーン上の FF 数の最大値を表す．n 本のスキャンチェーンで，全てのスキャンチェーン上の FF 数をほぼ同数とした場合，

$$max\_FF = \lceil \frac{\#FF}{n} \rceil$$

となる．$\#FF$ は総フリップフロップ数を表す．多重スキャンチェーンは，テスト実行時間削減に非常に有効であるが，テストデータ量は不変であり，スキャン入出力ピン数は増加する．テストデータ量およびテスト実行時間の削減のために，多重スキャンチェーンに対して，1 本の外部入力ピンからスキャンインデータを入力する手法が提案されている[42, 43, 44]．また多重スキャンチェーンの構成法として，スキャンチェーンを異なる点で分岐させ，ツリー状に構成する手法もある[45]．

複数のスキャンチェーンからのデータを圧縮するためにしばしば MISR が用いられることがある．しかしその場合には，面積オーバーヘッドが問題となる．そこで，スキャン出力が 1 本となるように並列

スキャンチェーンを構成する手法がある．文献 [46] の手法では，スキャン入力とスキャン出力がともに 1 本で，スキャンチェーンの内部で部分的に並列に FF が配置された構造のスキャンチェーンが提案されている．ここでは，内部が部分的に並列化できる条件をいくつか示しており，例えば，1 つの信号線から分岐した複数の枝に接続した FF や，同じゲートの入力線に接続した FF は並列化することができる．

一般的に，多重スキャンチェーンでは，各スキャンチェーン上に配置する FF 数をほぼ同数にし，スキャンチェーン上の FF 数の最大値を最小にした場合に，最もテスト実行時間が短縮されるが，同数でない場合にテスト実行時間が最小となる場合がある．例えば，一つの回路が複数の論理ブロックに分割され，各論理ブロックのテストベクトル数とスキャン FF 数が異なる場合，各スキャンチェーン上の FF 数によって，テスト実行時間が異なる．文献 [47, 48] では，テストベクトル数の多い論理ブロックに接続するスキャンレジスタに対しては，FF 数が少なくなるように多重スキャンチェーン構成することで，テスト実行時間の短縮を実現している．

5. まとめ

本論文では，近年発表された論理回路に対するテスト時間削減法について概説した．内容としては，組合せ回路と順序回路に対するテストコンパクション法，およびスキャン回路に対するテスト実行時間削減法について説明した．

今後 VlSI の大規模化が進むにつれ，テストコスト削減技術の重要性は一層高くなると考えられる．さらに回路の複雑化に伴い，遅延故障，ブリッジ故障，クロストーク故障，オープン故障などの縮退故障以外の故障モデルに対するテストベクトル数削減の手法を開発する必要がある．また，故障診断や高位レベルでのテストなどに対しても，テストベクトル数削減やテスト実行時間削減のための手法の研究開発が望まれる．

謝辞

本研究は一部，日本学術振興会の科学研究費補助金 (課題番号 15500043, 45300021) および栢森情報科学振興財団研究助成金（交付番号 K14 研 VII 第 142 号）の助成を得た．

参考文献

1) P. Goel, and B.C. Rosales, “Test generation and dynamic compaction of tests,” Proc. Int. Test Conf., pp.189–192, 1979.

2) 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,” IEEE Trans. on Computer-Aided Design, pp.1496–1504, Dec. 1995.

3) M.H. Schulz, E. Trischler, and T.M. Sarfert, “SOCRATES: A highly efficient automatic test pattern generation system,” IEEE Trans. on Computer-Aided Design, pp.126–137, Jan. 1988.

4) I. Hamzaoglu, and J. Patel, “Test set compaction algorithms for combinational circuits,” IEEE Trans. on Computer-Aided Design, vol.19, no.8, pp.957–963, 2000.

5) X. Lin, J. Rajski, I. Pomeranz, and S.M. Reddy, “In static test compaction and test pattern ordering for scan designs,” Proc. Int. Test Conf., pp.1088–1097, 2001.

6) 宮崎慎二，梶原誠司，“二重検出法に基づく故障シミュレーションの高速化について，” 信学技報 FTS2001-43，pp.43–48，2001.

7) 梶原誠司，イリスポメランツ，スダーカM．レディ，“トランジション故障に対するテストパターンの極小化手法について，” 信学技報 FTS98-126，pp.25–30，1999.

8) C.A. Balakrishnan, and V.D. Agrawal, “An exact algorithm for selecting partial scan flip-flops,” Journal of Electronic Testing: Theory and Application, vol.7, pp.83–94, 1995.

9) I. Pomeranz, L.N. Reddy, and S. M.Reddy, “Compactest: A method to generate compact test sets for combinational circuits,” Proc. of the lnt. Test Conf, pp.194–203, Oct. 1991.

10) P. Goel, “An implicit enumeration algorithm to generate tests for combinational logic circuits,” IEEE Trans. on Computers, vol.C-30, no.3, pp.215–222, 1981.

11) J.P. Roth, “Diagnosis of automata failures: A calculus and a method,” IBM J. Res. Develop., pp.278–291, 1966.

12) S.B. Akers, and B. Krishnamurthy, “On the application of test counting to VLSI testing,” Technical Report No. CR85-12, Computer Research Lab., Tektronix Laboratories, 1985.

13) M. Geuzebroek, J. der Linden, and A. van de Goor, “Test point insertion for compact test sets,” Proc. Int. Test Conf., pp.292–301, 2000.

14) T. Hosokawa, M. Yoshimura, and M. Ohta, “Novel DFT strategies using full/partial scan designs and test point insertion to reduce test application time,” IEICE Trans. on Fundamentals, vol.E84-A, no.11, pp.2722–2730, 2001.

15) M. Yoshimura, T. Hosokawa, and M. Ohta, “A test point insertion method to reduce the number of test patterns,” Proc. Asian Test Symp., pp.298–304, 2002.

16) M. Nakao, S. Kobayashi, K. Hatayama, K. Iijima, and S. Terada, “Low overhead test point insertion for scan-based BIST,” Proc. Int. Test Conf., pp.348–357, 1999.

17) M.S. Hsiao, E.M. Rundnick, and J.H. Patel, “Fast algorithms for static compaction of sequential circuit test vectors,” Proc. VLSI Test Symp., pp.188–195, 1997.

18) M.S. Hsiao, E.M. Rundnick, and J.H. Patel, “Fast static compaction algorithms for sequential circuit test vectors,” IEEE Trans. on Computer, vol.8, pp.311–322, 1999.

19) 樋上喜信，高松雄三，樹下行三，“リセット機能を持つ順序回路に対するテスト系列圧縮法，” 情報処理学会論文誌，vol.42, no.4，pp.1036–1044，2001.

20) T.M. Niermann, R.T. Roy, J.H. Patel, and J.A. Abraham, “Test compaction for sequential circuits,” IEEE Trans. Computer-Aided Design, vol.2, no.2, pp.260–267, 1992.

21) F. Corno, P. Prinetto, M. Rebaudengo, and M.S. Reorda, “New static compaction techniques of test sequences for sequential circuits,” Proc. European Design and Test Conf., pp.37–43, 1997.

22) 細川利典，井上智生，平岡敏洋，藤原秀雄，“時間展開モデルを用いた無閉路順序回路のテスト系列圧縮方法,” 電子情報通信学会論文誌 D-1，vol.J82-D-1, no.7，pp.869–878，1999.

23) I. Pomeranz, and S.M. Reddy, “On static compaction of test sequences for synchronous sequential circuits,” Proc. Design Automation Conf., pp.215–220, 1996.

24) I. Pomeranz, and S.M. Reddy, “An approach for improving the levels of compaction achieved by vector omission,” Poc. Int. Conf. Computer-Aided Design, pp.463–466, 1999.

25) R. Guo, I. Pomeranz, and S.M. Reddy, “Procedures for static compaction of test sequences for synchronous sequential circuits based on vector restoration,” Proc. Design Automation and Test in Europe, pp.583–587, 1998.

26) R. Guo, I. Pomeranz, and S.M. Reddy, “On speed-up vector restoration based static compaction of test sequences for sequential circuits,” Proc. Asian Test Sympo., pp.467–471, Dec. 1998.

27) I. Pomeranz, and S.M. Reddy, “Vector restoration based static compaction of test sequences for synchronous sequential circuits,” Proc. Int. Conf. Computer Design, pp.360–365, 1997.

28) M.S. Hsiao, and S.T. Chakradhar, “Partitioning and reordering techniques for static test sequence compaction of sequential circuits,” Proc. Asian Test Sympo., pp.452–457, 1998.

29) T.J. Lambert, and K.K. Saluja, “Methods for dynamic test vector compaction in sequential test generation,” Proc. Int. Conf. VLSI Design, pp.166–169, 1996.

30) E.M. Rundnick, and J.H. Patel, “Efficient techniques for dynamic test sequence compaction,” IEEE Trans. on Computer, vol.8, pp.323–330, 1999.

31) A. Raghunathan, and S.T. Chakradhar, “Acceralation techniques for dynamic vector compaction,” Proc. Int. Conf. Computer-Aided Design, pp.310–317, 1995.

32) I. Pomeranz, and S.M. Reddy, “Dynamic test compaction for synchronous sequential circuits using static compaction techniques,” Proc. Int. Symp. Fault-Tolerant Computing, pp.53–61, 1996.

33) 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.

34) P.C. Chen, B.D. Liu, and J.F. Wang, “Overall consideration of scan design and test generation,” Proc. Int’l Conf. on CA, pp.9–12, Nov. 1992.

35) H. Fujiwara, and A. Yamamoto, “Parity-scan design to reduce the cost of test application,” IEEE Trans. on Computer Aided Design, vol.12, no.10, pp.1604–1611, 1993.

36) J.S. Chang, and C.S. Lin, “A test clock reduction method for scan-desinged circuits,” Proc. Int’l Test Conf., pp.331–339, 1994.

37) 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, pp.2010–2016, 1994.

38) S.Y. Lee, and K.K. Saluja, “An algorithm to reduce test application time in full scan designs,” Dig. Int. Conf. on Computer-Aided Design, pp.17–20, Nov 1992.

39) S.Y. Lee, and K.K. Saluja, “Sequential test generation with reduced test clocks for scan designs,” Proc. VLSI Test Symp., pp.220–225, May 1994.

40) I. Pomeranz, and S.M. Reddy, “Static test compaction for scan-based designs to reduce test application time,” Proc. Asian Test Sympo., pp.198–203, Dec. 1998.

41) I. Pomeranz, and S.M. Reddy, “An approach to test compactin for scan circuits that enhances at-speed testing,” Proc. Design Automation Conf., 2001.

42) I. Hamzaoglu, and J.H. Patel, “Reducing test application tiem for full scan embeded cores,” Proc. Int. Symp. on Fault-Tolerant Comp., pp.260–267, June 1999.

43) F.F. Hsu, K.M. Butler, and J.H. Patel, “A case study on the implementation of the Illiois scan architecture,” Proc. Int. Test Conf., pp.538–547, Oct. 2001.

44) K.J. Lee, J.J. Chen, and C.H. Huang, “Using single input to support multiple scan chains,” Dig. Int. Conf. on Computer-Aided Design, pp.74–78, Nov. 1998.

45) 宮瀬紘平，梶原誠司，S.M. Reddy，“スキャンツリーを用いたテストデータ量最小化について，” 第48回FTC研究会資料，Jan. 2003.

46) Y. Higami, and K. Kinoshita, “Design of partially parallel scan chain,” Proc. European Design and Test Conf., p.626, Mar. 1997.

47) R. Gupta, and M.A. Breuer, “Ordering storage elements in a single scan chain,” Proc. Int’l Conf. on CA, pp.408–411, Nov. 1991.

48) S. Narayanan, C. Njinda, and M. Breuer, “Optimal sequencing of scan registers,” Proc. Int’l Test Conf, pp.293–302, Sep. 1992.