サーベイ論文

論理回路に対するテストコスト削減法——テストデータ量及びテスト実行時間の削減——

樋上　喜信[†]　　梶原　誠司[††]　　市原　英行[†††]　　高松　雄三[†]

Test Cost Reduction for Logic Circuits——Reduction of Test Data Volume and Test Application Time——

Yoshinobu HIGAMI[†], Seiji KAJIHARA[††], Hideyuki ICHIHARA[†††], and Yuzo TAKAMATSU[†]

あらまし　論理回路の大規模化とともに，テストコストの増大が深刻な問題となっている．特に大規模な論理回路では，テストデータ量やテスト実行時間の削減が，テストコスト削減の重要な課題である．本論文では，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ためのテストコンパクション技術，付加ハードウェアによるテストデータの展開・伸長を前提に圧縮を行うテストコンプレッション技術，及び，スキャン設計回路におけるテスト実行時間削減技術について概説する．

キーワード　論理回路，テストコスト，テストコンパクション，テストコンプレッション，テスト実行時間削減

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加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 年間でその技術は急速に進歩しており，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，テスト圧縮法と呼ばれている．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 2 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(注1)．

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(注2)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．テスト生成の中で用いられるため，LSI テスタや回路に付加的な負担を必要としないが，テスト生成時間は増加する．

一方，テストコンプレッションは，0，1 の 2 値で表されたテストパターンの表現方法を変換し，テスタに格納するテストデータ量を削減する手法である．テストデータはチップに埋め込まれた付加回路によりもとのテストパターンに展開してから，被テスト回路に印加される．チップ面積は増加するが，テストデータの削減率は高い．展開器としては，線形フィードバックシフトレジスタ（LFSR;Linear Feedback Shift Register），組合せ回路，有限状態機械を実現した順序回路，プロセッサなどが用いられる．

† 愛媛大学工学部情報工学科，松山市
Department of Computer Science, Faculty of Engineering, Ehime University, 3 Bunkyo-cho, Matsuyama-shi, 790–8577 Japan
†† 九州工業大学情報工学部電子情報工学科，飯塚市
Department of Computer Science and Electronics, Kyushu Institute of Technology, 680–4 Kawazu, Iizuka-shi, 820–8502 Japan
††† 広島市立大学情報科学部情報機械システム工学科，広島市
Faculty of Information Sciences, Hiroshima City University, 3–4–1 Otsuka-higashi, Asaminami, Hiroshima-shi, 731–3194 Japan

（注1）：英語では二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．
（注2）：本論文では，テストベクトルの集合をテストパターンと呼ぶ．

テストコンパクションとテストコンプレッションはどちらもテスト圧縮の技術であるが，テストコンパクションは圧縮したデータを圧縮前のデータに復元できない技術で，テストコンプレッションは圧縮したデータを圧縮前のデータに復元できる技術ということもできる．これらは両立する技術であり，テストコンパクション技術を使って生成したテストパターンをテストコンプレッションにより更にそのデータ量を削減するなど，組み合わせて用いればテストデータ量は劇的に削減される．

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は，同時にテスト実行時間削減をも意味する．しかしながら，スキャン回路のテストや IDDQ テスト環境などでは，テストベクトル数最小を目指したテストコンパクションとは異なる手法によって，テスト実行時間が削減される場合もある．

本論文では，テストコンパクションとテストコンプレッションの 2 通りのテスト圧縮法，及びテスト実行時間削減法について概説する．テストコンパクションについては，論理テスト環境における，論理回路の単一縮退故障のテスト生成で用いられる手法，及び IDDQ テスト環境を前提とした手法を述べる．テストコンプレッションについては，付加回路の機能に着目して分類し，テストデータ量削減のための手法を説明する．更にテスト実行時間削減法に関して，論理テスト環境におけるスキャン回路に対する手法について，単一スキャンチェーンの場合と多重スキャンチェーンの場合に分類して説明する．

本論文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論理テスト環境における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 IDDQ テスト環境におけるテストコンパクションについて述べる．更に，**5.** では，テストコンプレッションについて，**6.** では，論理テスト環境における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 **7.** で本論文のまとめと今後の課題について述べる．

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，図 1 のようないったん生成したテストパターンを小さくするための手法と，図 2 のような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．本章では，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．

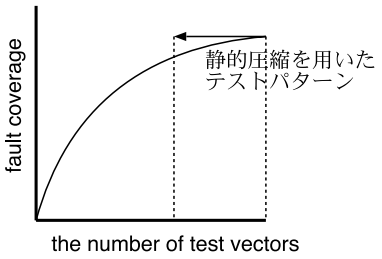

図 1　生成したテストベクトル数の削減
Fig. 1　Reduction of the number of generated test vectors.

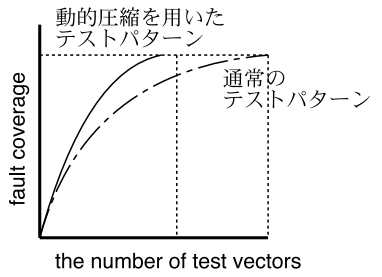

図 2　小さなテストパターンの生成
Fig. 2　Generation of a small size of a test pattern.

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，及び，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(注3)．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．例えば，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

(注3)：静的圧縮という用語は広義には，いったん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
Table 1　An example of a test pattern.

(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

とすると，これら二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．例えば，ある回路に対するテストパターンが，表 1(a) のように $t_1, t_2, \ldots, t_5$ 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．例えば，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]．例えば，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，更にテストベクトル数を少なくできる [2], [4], [8]．例えば，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 $\{t_2, t_6\}$ という更に小さなテストパターンができる．テストベクトルの置換えは，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

いったん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，更に，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(注4)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

(注4)：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．

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]〜[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 2 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個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(注5)．この節で説明する手法は，すべてテストベクトル数削減を目的としている．

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もとのテスト系列のものと変化しないこと，あるいは，状態が変化してももと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もと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もと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換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もとの故障検出率が保証される場合がある．そのような手法の一つは，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，各故障が顕在化及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

(注5)：本章では広義の静的圧縮法，及び広義の動的圧縮法を述べる．

くなる [19], [20].

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 [21], [22] では，二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ラップして二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，もと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，大きなテスト系列長短縮の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 [23] 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 [24], [25] 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 [26], [27]，テストベクトルを復帰させる手法（vector restoration）[28]〜[30] 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すべ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．この手法をもとに，計算時間短縮のための手法 [31]〜[33] や，テスト生成法に応用した手法 [34], [35] などが提案されている．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．

もと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．例えば与えられたテスト系列を 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 [36] では，もとのテスト系列 T を T_1, T_2 の二つの系列に分割した後，それらを並べ換えることによって，$T' = T_2 - T_1$ 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，もとのテスト系列 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 [26]．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，一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 [37] 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，高い効果を上げている [38]〜[40]．更にクロックを停止させ，状態を固定した後，外部入力を変更することも有効である [41].

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 [42], [43].

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 [44] 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. IDDQ テストのためのテストコンパクション

CMOS 回路において，静的電源電流を測定することによって故障回路を識別する，IDDQ テストがある [45]〜[48]．IDDQ テストでは，論理テストと異なり，故障の影響を外部出力まで伝搬させる必要がないため，テスト生成が容易である．また，論理テストでは検出できない故障を検出できるなどの利点をもつ．反面，静的電源電流を測定するために，回路が安定状態に達するまで待つ必要があり，テスト実行時間が長くなるという欠点がある．したがって，テスト実行時間短縮が特に重要な課題となっている．

ここでは，テストベクトルを，IDDQ を観測する必要のあるベクトル（IDDQ 観測ベクトルと呼ぶ）と IDDQ を観測する必要のないベクトル（状態遷移ベクトルと呼ぶ）の 2 種類に分類して考える．組合せ回路の場合，状態遷移ベクトルを印加する必要はなく，すべてのテストベクトルが IDDQ 観測ベクトルとなる．順序回路においては，未検出故障を新たに検出しないが，別の故障を顕在化するために回路を特定の状態に遷移させるようなテスト系列を印加する必要がある．

IDDQ テストにおけるテストコンパクションでは，メモリ記憶容量などテストデータ量削減を主な目的とした場合，テストベクトル数全体を削減することが重要となり，テスト実行時間短縮を主な目的とした場合には，テスト実行時間に大きな影響を与える，IDDQ 観測ベクトルの削減が重要となる．

以下 **4. 1** では，組合せ回路のテストベクトル数削減，**4. 2** では順序回路のテストベクトル数削減，**4. 3** では順序回路の IDDQ 観測ベクトル数削減について述べる．**4. 1** と **4. 2** で述べる手法は，テストベクトル数全体を削減することが目的であり，**4. 3** で述べる手法は，テスト実行時間の削減が目的である．

4. 1 組合せ回路のテストベクトル数削減

論理テスト環境でのテストコンパクションと同様，一つのテストベクトルでできるだけ多くの故障を検出するようにテスト生成を行うことで，テストベクトル数を削減することができる．IDDQ テストのテスト生成では，故障を顕在化させるだけでよく，論理テストのように故障の影響を伝搬する必要がなく，テスト生成は容易である．したがって，縮退故障を対象とした場合には，論理テストと同様の手法が適用可能で，テストコンパクションも容易である．しかしながら，IDDQ テスト環境では，ブリッジ故障モデルが用いられることが多く，縮退故障に対するものとは異なる手法が用いられる．回路内で起こり得るブリッジ故障は，レイアウト情報により求めることが可能であるが，レイアウト情報が利用できない場合には，すべての信号線間のブリッジ故障を対象とする必要がある．そのような場合には，対象故障数が膨大となるため，論理テストに対するテストコンパクションで用いられたような手法をそのまま適用することが困難となる．例えば，故障ひとつひとつを個別に扱った手法や，すべての故障に対して故障検出の情報（どのテストベクトルで検出されたかなど）を記憶する手法などは，適用できない場合がある．そこで，特定の故障を対象にしない手法や，複数の故障を同時に対象とした手法が提案されている．

テスト生成時に適当な論理回路を付加し，特定の信号線の組に 0 または 1 を割り当てることで，多くのブリッジ故障を同時に検出するテストベクトルを生成する手法が提案されている [49]．また，ランダムベクトルをもとに，ビット値の変更によりそのベクトルが検出する故障数を最大化し，テストベクトル数を抑制する手法もある [50], [51]．この手法は，実装が比較的容易で圧縮の効果も高いが，回路規模の増大とともに計算時間の増大が問題となる．

組合せ回路のブリッジ故障に対して，テスト生成後にテストベクトルを削減する手法として，必須故障に着目した手法が提案されている [52]．必須故障とは，生成されたテストパターン中の一つのテストベクトルでしか検出されない故障のことである．ここでは，必須故障の検出に着目してテストベクトルを変更し，新たに冗長となったテストベクトルを削除している．

レイアウト情報などを用いれば対象故障数が少なくなり，すべての故障に対する故障検出の情報を利用して，テストベクトルを削減することが可能となる．文献 [53] では，すべての故障に対して，どのテストベクトルで検出されたかを記憶したビットアレーを用いて，近似的な最小数のテストベクトルを選択することを可能にしている．

4.2 順序回路のテストベクトル数削減

順序回路に対する手法においても，回路内のすべての信号線間のブリッジ故障を扱う際には，計算時間や計算機のメモリ容量を実用的な範囲内に抑えることが重要となる．

回路内のすべての信号線とゲート内部ノード間のブリッジ故障を対象に，部分テスト系列の削除や置換を行う手法が提案されている [54], [55]．この手法では，与えられたテスト系列を，故障検出に着目して複数の部分テスト系列に分割することと，状態遷移に着目して部分系列の削除や変更を行うことによって，計算時間の増大を防いでいる．また，テストベクトル数全体を少なくするテスト生成法として，遺伝的アルゴリズムを用いた手法が提案されている [56]．この手法では，故障検出率が最大で，テスト系列長が最短となるような適合度関数を用いることによって，短いテスト系列の生成を実現している．

4.3 順序回路の IDDQ 観測ベクトル数削減

IDDQ テストでは，IDDQ を観測するために長い時間を必要とするため，テスト実行時間短縮のためには，IDDQ 観測ベクトルを減らすことが重要である．組合せ回路においては，テストベクトル数全体を削減することと，IDDQ 観測ベクトル数を削減することは，同じことを意味するが，順序回路においては，未検出故障を一つも検出しないテストベクトルであっても，印加しなければならない場合があり，テストベクトル数全体を削減する手法と，IDDQ 観測ベクトル数を削減する手法は異なるものとなる．

ブリッジ故障を対象に，与えられたテスト系列の中から IDDQ 観測ベクトルを選択する手法として，最も単純なのは，故障シミュレーションを行い，未検出故障を少なくとも一つ検出するテストベクトルを IDDQ 観測ベクトルとして選択する手法である [57], [58]．この手法では，各故障に対して，テスト系列中で初めて検出されるテストベクトルを IDDQ 観測ベクトルとして選択するため，正常回路の信号値を計算するだけでよく，計算量が故障数に影響を受けない．

最小数の IDDQ 観測ベクトルを選択するためには，各故障が検出されるテストベクトルすべてについて考慮する必要があり，これを実現するためには，各故障について故障影響の伝搬を考慮したシミュレーションが必要で，ブリッジ故障のように故障数が多い場合には，計算時間やメモリ量を削減するための工夫が必要となる．文献 [59] では，ある一定数以上のブリッジ故障を検出するテストベクトルを無条件に IDDQ 観測ベクトルとして選択し，そのテストベクトルで検出されない故障についてのみ，故障影響の伝搬を考慮したシミュレーションを行うことで，計算時間の増大を防いでいる．また文献 [60] では，複数のブリッジ故障影響の伝搬を同時に考慮したシミュレーションを行い，計算時間の短縮を実現している．

縮退故障に対して，少ない数の IDDQ 観測ベクトルを選択する手法が文献 [61] で提案されている．この手法においても，未検出故障の中で最も多くの故障を検出するテストベクトルを優先的に選択することで，少ない数の IDDQ 観測ベクトルを実現している．

5. テストコンプレッション

テストコンプレッションは，図 3 に示すように，テストパターン T_D をデータ量の少ないテストデータ T_E に変換してチップに入力する手法である．T_E は，被テスト回路に印加される前に，チップ内に組み込んだ付加回路を通じてもとの T_D に展開される．したがって，テストコンプレッションは，テストパターンの一部をハードウェア化することで，テストデータ量を少なくしていると考えることができる．付加回路によるチップ面積の増加と引換えにテスタに必要なテストパターンの記憶容量と，テスタからテスト対象回路までのテストパターンの伝送時間を削減することができる．以下では，テストデータ量削減の手法について，LFSR を用いた手法，テストデータ符号化の手法に分類して説明する．

5.1 LFSR を用いたテストデータ量削減

LFSR は擬似乱数発生回路として，特に，論理回路の BIST（Built-In Self Test：組込み自己テスト）方式のテストにおけるテストパターン発生回路として，古くから知られている．擬似乱数によるテストパター

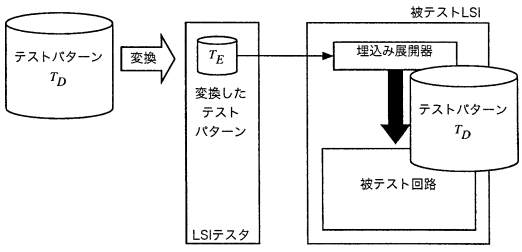

図 3　テストコンプレッションの概要
Fig. 3　An overview of test compression.

ンで多くの故障が検出できる一方で，それでは検出困難な故障も存在する．そこで，十分な故障検出率を確保するため，検出困難な故障を検出するテストベクトルがより確実に発生できるような工夫もなされている．

代表的な手法に，擬似乱数によるテストベクトルの中に検出困難な故障を検出するものが含まれるよう，擬似乱数発生途中に LFSR の値を強制的に変更する，リシーディング（reseeding）と呼ばれる手法がある [62]〜[67]．LFSR で発生する乱数系列はその初期値（シード）に依存して異なる系列が得られることに着目しており，検出困難な故障を検出するテストベクトルの発生を伴うシード複数個をあらかじめ計算し，外部の LSI テスタ（またはチップ上のメモリ）から与えることで，高い故障検出率を達成できる．一般に多くの故障は擬似乱数で検出できるため，シードのデータ量は，通常のテスト生成で求めるテストパターンよりも大幅に少なくなる．

リシーディングは LFSR の入力を変更する手法であるが，LFSR から出力される擬似乱数を変換する BAST（BIST-Aided Scan Test）手法 [68] も提案されている．テスト生成により求めたテストベクトルが実現されるように擬似乱数の一部のビットを変換することで，テスト生成で得られる故障検出率を保証するテストパターンが発生できる．ある故障に対して生成したテストベクトルに未設定の入力値が多く含まれることを利用すれば，変換すべきビットの数は少なくてすむ．

LFSR を利用したテストコンプレッションでは，テストパターンの大部分はハードウェアとして実装しているため，すべてのテストパターンをテスト生成により求める場合より，テストに必要なデータ量は少なくなる．一方，リシーディングに必要なシードの計算に時間がかかるため，テスト生成に必要な時間はすべてのテストパターンを求める場合より大幅に増加する場合がある．また，設計時の検証パターンを用いて機能テストを行う場合のように，あらかじめ与えられているテストパターンを用いる場合には，本質的に LFSR のリシーディングでは対応できない．これに対して，次に述べる符号化法を用いた手法は，基本的に任意のテストパターンに対して適応可能な手法である．

5.2 テストデータ符号化法

テストデータ符号化法は，与えられたテストパターンを符号化することで，テストデータ量を削減する手法である．これまで様々な符号化法が提案されていて，大きなデータ量削減の効果があると報告されている．ここでは，符号化したテストパターンを復号するための復号器の種類によって，提案されている手法を (1) 組合せ回路によって復号を行う手法，(2) 順序回路によって復号を行う手法，(3) プロセッサによって復号を行う手法の，3 種類に分類して紹介する．一般に，(1) から (3) に進むに従って，ハードウェアオーバヘッドは増すものの，データ量が削減される割合は高くなると考えられる．一方で，テストパターンを生成した後でなければ，復号回路を合成できないため，設計への負担が増加する．

5.2.1 組合せ回路によって復号を行う手法

組合せ回路による復号を行う手法は，図 4 に示すように，外部入力数 N より少ない出力数 M をもつ組合せ回路をテスタと被テスト回路の間に置くことで，両者をつなぐ信号線数の削減（width compression）に利用されている [69]〜[72]．復号器の組合せ回路に入力されたビット幅 N の符号語は瞬時にビット幅 M のベクトルに復号され，被テスト回路に印加される．

文献 [69], [70] では，BIST におけるチャネル幅削減の手法が提案されている．提案手法では，被テスト回路の外部入力間のコンパチビリティを調べ，復号器（組合せ回路）を LFSR と被テスト回路の間に設置する手法を提案している．この回路を用いることにより LFSR のサイズを小さくできると報告されている．

外部テストにおいて多重スキャン設計に対して有効なテストコンプレッション法も提案されている [71], [72]．多重スキャン設計のスキャンチェーン本数に対して，スキャン入力数を減らすことで，テストデータ量を削減する．文献 [71] では，テストベクトルの未設定入力を利用して，EXOR ゲートで構成される線形回路で復号できるようなテストコンプレッション法が提案されている．文献 [72] では，スキャンチェーンへの入力ベクトルの種類（ビットパターン）が多くないことに着目して，スキャン入力数を減らす手法が提案されている．

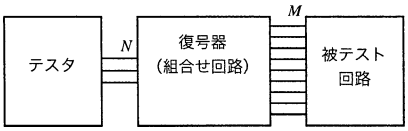

図 4 組合せ回路を用いた入力数圧縮
Fig. 4 Test width compression using combinational circuits.

組合せ回路による復号を用いた手法では，テスタと被テスト回路間の信号線数は減らせるものの，テストパターンの印加に必要な時間を減らすことができない．これに対して，この後に述べる二つの手法は主にテスト実行時間を削減することを目的としている．

5.2.2 順序回路によって復号を行う手法

順序回路による復号を行う手法として，統計型符号を用いたもの [73]〜[78] とランレングス符号を用いたもの [79]〜[83] の 2 種類を紹介する．これらの符号化法では，組合せ回路を用いて復号する場合と異なり，符号語長や復号後のベクトル長が可変であるため，順序回路を用いて入出力のタイミングを調整する必要がある．

統計型符号は，データ中のブロックの出現頻度が高いものには短い符号語を，出現頻度の低いものには長い符号語を割り当てる符号化法である．文献 [73] では一つのテストベクトルを一つの固定長ブロックと考え，統計型符号であるハフマン符号とコンマ符号で符号化を行う手法が提案されている．この手法では，テスト系列中のテストベクトルの出現回数を調べ，ハフマン符号またはコンマ符号で符号化する．非スキャンの順序回路を対象とした場合には，一般的にテストパターン中に同じテストベクトルが複数含まれるため，テストデータ量削減の効果が大きい．

表 2 に 4 入力のテスト系列に対するハフマン符号とコンマ符号の例を示す．この例では，80 個のテストベクトルからなるテストパターンに対して，4 種類のユニークなテストベクトルが現れる場合を示している．表中の出現回数は 4 種類のベクトルが現れる回数を示し，出現率は出現回数の割合を示している．ハフマン符号では，ハフマンのアルゴリズムに従って，出現率が高いベクトルにはなるべく短い符号語を，出現率が低いベクトルには長い符号語を割り当てる．コンマ符号では，0 を符号語の切れ目を表す文字として用い，最もよく現れるパターンに符号語 0，次に出現率の高いパターンに符号語 10，次に出現率の高いパターンに符号語 110，というように 1 ビットずつ長い符号を割り当てる．この例では，320 ビットのテストパターンが，ハフマン符号を用いた場合は 135 ビット，コンマ符号を用いた場合は 275 ビットに削減される．

表 2　ハフマン符号とコンマ符号
Table 2　Huffman code and comma code.

ベクトル	出現回数	出現率	ハフマン符号	コンマ符号
0011	45	0.5625	0	0
0100	15	0.1875	10	10
1010	15	0.1875	110	110
1111	5	0.0625	111	1110

統計型符号化によって得られた符号語は表 2 に示すようにその長さが符号語によって異なるため，テスタから 1 ビットずつ受け取って，符号語がそろった時点で復号する必要がある．そのため，復号器は最初に述べたように順序回路である必要がある．例として表 2 のハフマン符号に対する展開器（1 入力 4 出力の順序回路）の状態遷移図を図 5 に示す．

文献 [73] の手法は入力数が少ない被テスト回路のみを対象としており，入力数が大きくなると適応するのが困難となる．そこで，文献 [74] では，テストパターンを固定ビットのブロックに分割し，それぞれのブロックをハフマン符号化する手法を提案している．更に，符号化したテストパターンを復号するときに必要なデコーダのサイズを小さくすることも考えている．提案法では，得られたすべてのブロックを符号化するのではなく，出現確率の高いブロックを数個選んで符号化し，残りのブロックはそのままのパターンを符号語とする．これにより，すべてのブロックを符号化する場合に比べてテストデータ量削減の割合はわずかに落ちるものの，デコーダのサイズを小さく抑えることができる．

文献 [75]〜[78] では，統計型符号による符号化に適したテストパターンを生成するための手法が提案されている．これらの手法は，テストベクトル中の未設定入力値に 0 または 1 を割り当てることによって，少ないテストデータ量を実現している．

一方，ランレングス符号化はデータ中の連続した 0 または 1 の列（ラン）を，符号語に置き換える手法である．文献 [73], [79]〜[81] ではランレングス符号をテストコンプレッションに用いている．また文献 [82], [83]

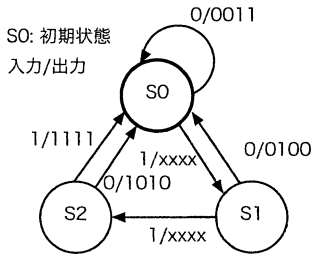

図 5　順序回路による復号
Fig. 5　Decompression using a sequential circuit.

では，ランレングス符号から派生したゴーロム符号やFDR（Frequency-Directed Run-length）符号を用いたテストコンプレッション法を提案している．ランレングス符号によるテストデータ削減の割合を高めるためには，テストパターン中になるべく長いランが現れることが必要となる．文献[79], [80]では，テストパターンに対してランレングス符号化の前処理として，Burrows-Wheeler変換を施すことで，テストパターン中のランの長さを長くし，テストデータ量削減の割合を高めている．

文献[81]では，符号化の対象のテストパターン中の0の数を増やせば，自然に0のランが増えると考えて，テスト系列の差分ベクトル系列を考える．テスト系列 $T_D = \{t_1, t_2, t_3, \ldots, t_n\}$ が与えられたとき，その差分ベクトル系列 T_{diff} は $T_{diff} = \{t_1, t_1 \oplus t_2, t_2 \oplus t_3, \ldots, t_{n-1} \oplus t_n\}$ と定義される．テスト系列は一般的に，一部の入力だけ変化する場合が多いため，もとのテスト系列に比べて差分ベクトル系列には0が増えることが期待できる．また，テスト系列内のベクトルの順序を入れ換えることが可能ならば，差分ベクトル系列中の0が増えるように順序を並べ換えることも行う．

更に文献[82]や[83]では，ランレングス符号の代わりにゴーロム符号やFDR符号を用いる手法を提案している．これらの符号語は，上述したランレングス符号と異なり，可変長の符号語であるため，より大きなテストデータ量削減を実現できる．また，デコーダが比較的簡単になるように符号語を構成する工夫も行っている．

これらの順序回路により復号を行う手法は，展開器のハードウェアオーバヘッドが大きいものが多い．しかしながら，そのオーバヘッドは回路規模にはほとんど依存しないため，今日の大規模回路においてはその影響は小さくなっている．統計型符号やランレングス符号を用いた圧縮では，符号語の長さや展開後のベクトル長が可変であるため，テスタと復号器間及び復号器と被テスト回路間のタイミングを調整することが必要である．文献[84]では，タイミングの調整を考慮したハフマン符号化法に基づく手法が提案されており，今後このようなテスト実行時のタイミングを考慮したテストコンプレッションの研究が必要になると思われる．

5.2.3 プロセッサによる復号を行う手法

文献[85]と文献[86]では，被テスト回路がプロセッサを実装していることを前提としたテストコンプレッション法を提案している．これらの手法では，復号アルゴリズムはソフトウェアとして実装され，プロセッサ上で復号処理が実行される．そのため，複雑な復号化処理を必要とするテストコンプレッションが可能となる．なお，テスト対象の回路はプロセッサ以外の部分であり，プロセッサ自体のテストについてはここでは対象としない．

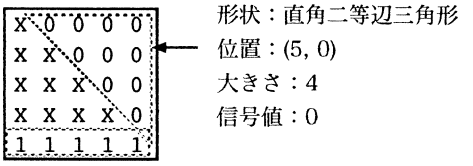

図6 基本図形の置換えによる圧縮例
Fig. 6 An example of geometric-primitives-based compression.

文献[85]では，テストベクトルを固定長のブロックに分割しておき，テストベクトル間のブロック単位の差分をとり，異なるブロックだけを符号化してプロセッサ用の内部メモリに記憶する手法を提案している．

文献[86]では，R 入力で C 個のテストベクトルからなるテストパターンを $R \times C$ ビットのマトリックスと考え，これをあらかじめ定義された基本形状に分割し，それぞれの形状の形と位置に従って符号化を行う．例えば，図6に示すような，5×5 ビットのテストパターンを符号化する場合，信号値1の直線と信号値0の三角形の二つの基本図形でテストパターンが構成されていると考え，それぞれの形状，位置，大きさ，信号値を表す符号語に置き換える．ブロックの左上を原点(0,0)と考えると，図6における信号値0の三角形は座標(5,0)から下を向いた1辺の長さが4の直角二等辺三角形であることを表す符号語で置き換える．

プロセッサによる復号を用いたテストコンプレッション法は，当然ながらプロセッサを搭載したLSIにしか適応できないため適応範囲は狭いものの，大きなテストデータ量削減が可能であるため，今後多くの研究が行われる可能性がある．

6. 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，

$$TAT = \#Vec + \#FF \times (\#Vec + 1) \qquad (1)$$

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 の値を設定及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．したがって，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．

6.1 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式 (1) 第 2 項は，すべてのテストベクトルに対して，すべての 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．したがって，テスト実行時間を短縮するための手法として，すべてのテストベクトルに対して，すべての 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(注6)．例えば，表 3 のような二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．

特別なハードウェア機構を用いて，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 [87] では，FF を二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，すべての FF をスキャンシフトする．また，文献 [88] 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ェア機構を提案している．例えば，あるモードでは一つの FF のみをスキャンシフトし，また別のモードでは，二つの 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 を順に一つずつ増加させる(注7)．

上記の二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ェア機構を付加する手法が提案されている [89], [90]．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．

特別なハードウェア機構を用いないで，一部の FF のみにデータを制御・観測する手法が提案されている [91], [92]．この手法では，各テストベクトルごとにスキャンシフトを行う FF 数を可変にしている．例えば，五つの FF が $FF_1, FF_2, \ldots, FF_5$ 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 [92] 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

表 3　テストベクトルの例
Table 3　An example of test vectors.

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

(注6)：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．

(注7)：これらの手法では，ハードウェ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，テスト時の動作及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．

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 [93] とパーシャルスキャン回路に対する手法 [94] が提案されている．n 個のスキャン FF がある場合，すべての 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 [95]．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．その後更に，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 [95] 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 [96]．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，スキャンイン及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，更にテスト実行時間を削減する．

6.2 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呼ぶ．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，

$$TAT = \#Vec + max\_FF \times (\#Vec + 1) \quad (2)$$

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，すべてのスキャンチェーン上の FF 数をほぼ同数とした場合，

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

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．テストデータ量及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 [97]〜[99]．図 7 に示されるように，一つの入力から，n 個の異なるスキャンチェーンに対して同じスキャンデータを入力する．異なる FF の値がテストベクトル中で同じ値をとったり，また未設定値をうまく利用すれば，故障検出率の低下を防ぐことができる．スキャンデータの観測に関しては，多重入力署名解析器（MISR）を用いることで，スキャンデータ観測用の外部出力ピン数を増大させることなく，故障影響の観測性の低下を防ぐことができる．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，トリー状に構成する手法もある [100]．

スキャンデータを観測するために MISR を必要とせず，1 本のスキャン出力でスキャンデータを観測する手法がある．文献 [101] 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，一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 数を同数にした場合に，最

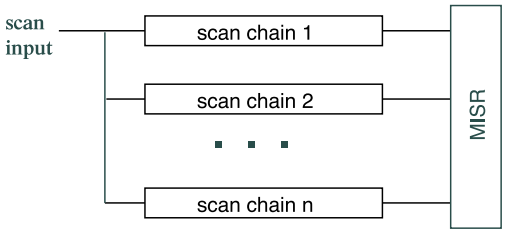

図 7　1 入力多重スキャンチェーン
Fig. 7　Multiple scan chain with a single input.

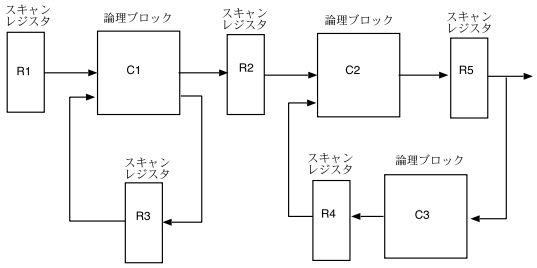

図 8　複数ブロックからなる回路例
Fig. 8　A circuit consisting of some blocks.

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．例えば図 8 の回路のように，各論理ブロックのテストベクトル数とスキャンレジスタ数が異なる場合，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 [102], [103] 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．

7. むすび

本論文では，近年発表された論理回路に対するテストコスト削減法について概説した．内容としては，組合せ回路及び順序回路に対するテストコンパクション法，IDDQ テスト環境におけるテストコンパクション法，テストコンプレッション法，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いっそう高くなると考えられる．更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やテストデータ量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データ量削減，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）及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．

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. Comput.-Aided Des. Integr. Circuits Syst., vol.14, no.12, 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. Comput.-Aided Des. Integr. Circuits Syst., vol.7, no.1, pp.126–137, Jan. 1988.

[4] I. Hamzaoglu and J.H. Patel, “Test set compaction algorithms for combinational circuits,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., 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, 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, 1999.

[8] C.A. Balakrishnan and V.D. Agrawal, “An exact algorithm for selecting partial scan flip-flops,” J. Electron. Test.: Theory Appl., 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. lnt. Test Conf., pp.194–203, Oct. 1991.

[10] P. Goel, “An implicit enumeration algorithm to generate tests for combinational logic circuits,” IEEE Trans. Comput., 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. Fundamentals, vol.E84-A, no.11, pp.2722–2730, Nov. 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. Comput., vol.8, no.3, pp.311–322, 1999.

[19] Y. Higami, Y. Takamatsu, and K. Kinoshita, “Test sequence compaction for sequential circuits with reset states,” Proc. Asian Test Symp., pp.165–170, Dec. 2000.

[20] 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，” 情処学論，vol.42, no.4, pp.1036–1044, 2001.

[21] R.T. Roy, T.M. Niermann, J.H. Patel, A. Abraham, and R.A. Saleh, “Compaction of ATPG-generated test sequences for sequential circuits,” Dig. Int. Conf. on Computer-Aided Design, pp.382–385, 1988.

[22] T.M. Niermann, R.T. Roy, J.H. Patel, and J.A. Abraham, “Test compaction for sequential circuits,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.2, no.2, pp.260–267, 1992.

[23] 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.

[24] 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，” 信学論（D-I），vol.J82-D-I, no.7, pp.869–878, July 1999.

[25] T. Hosokawa, T. Inoue, T. Hiraoka, and H. Fujiwara, “Static and dynamic test sequence compaction methods for acyclic sequential circuits using a time expansion model,” Proc. Asian Test Symp., pp.192–199, 1999.

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

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

[28] 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.

[29] 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 Symp., pp.467–471, Dec. 1998.

[30] 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.

[31] S. Bommu, S.T. Chakradhar, and K.B. Doreswamy, “Static test sequence compaction based on segment reordering and accelerated vector restoration,” Proc. Int. Test Conf., pp.954–961, 1998.

[32] S. Bommu, S.T. Chakradhar, and K.B. Doreswamy, “Static compaction using overlapped restoration and segment pruning,” Proc. Int. Conf. Computer-Aided Design, pp.140–146, 1998.

[33] S. Bommu, S.T. Chakradhar, and K.B. Doreswamy, “Vector restoration using accelerated validation and refinement,” Proc. Asian Test Symp., pp.458–466, 1998.

[34] R. Guo, I. Pomeranz, and S.M. Reddy, “A fault simulation based test pattern generation for synchronus sequential circuits,” Proc. VLSI Test Symp., pp.260–267, 1999.

[35] R. Guo, I. Pomeranz, and S.M. Reddy, “Proptest: A property based test pattern generation for sequential circuits using test compaction,” Proc. Design Automation Conf., pp.653–659, 1999.

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

[37] 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.

[38] E.M. Rundnick and J.H. Patel, “Simulation-based techniques for dynamic test sequence compaction,” Proc. Int. Conf. Computer-Aided Design, pp.67–73, 1996.

[39] E.M. Rundnick and J.H. Patel, “Putting the squeeze on test sequences,” Proc. Int. Test Conf., pp.723–732, 1997.

[40] E.M. Rundnick and J.H. Patel, “Efficient techniques for dynamic test sequence compaction,” IEEE Trans. Comput., vol.8, no.3, pp.323–330, 1999.

[41] Y. Santosa, M. Merten, E.M. Rundnick, and M. Abramovici, “Freezeframe: Compact test generation using a frozen clock strategy,” Proc. European Design and Test Conf., pp.747–752, 1999.

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

[43] A. Raghunathan and S.T. Chakradhar, “Dynamic test sequence compaction for sequential circuit,” Proc. Int. Conf. VLSI Design, pp.170–173, 1996.

[44] 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.

[45] S. Chakravarty and P.J. Thadikaran, Introduction to

IDDQ Testing, Kluwer Academic Publishers, 1997.
[46] Y.K. Malaiya and R. Rajsuman, Bridging Faults and IDDQ Testing, IEEE Computer Society Press, 1992.
[47] R. Rajsuman, Iddq Testing for CMOS VLSI, Artech House Publisher, 1994.
[48] Y. Higami, Y. Takamatsu, K.K. Saluja, and K. Kinoshita, “Fault models and test generation for IDDQ testing,” Proc. Asia and South Pacific Design Automation Conf., pp.509–514, Jan. 2000.
[49] R.S. Reddy, I. Pomeranz, S.M. Reddy, and S. Kajihara, “Compact test generation for briging faults under IDDQ testing,” Proc. VLSI Test Symp., pp.310–316, April 1995.
[50] T. Shinogi and T. Hayashi, “A simple and efficient method for generating compact IDDQ test set for bridging faults,” Proc. VLSI Test Symp., pp.112–117, April 1998.
[51] T. Shinogi and T. Hayashi, “An iterative improvement method for generating compact tests for IDDQ testing of bridging faults,” IEICE Trans. Inf. & Syst., vol.E81-D, no.7, pp.682–688, July 1998.
[52] H. Kondo and K.T. Cheng, “An efficient compact test generator for IDDQ testing,” Proc. Asian Test Symp., pp.177–182, Nov. 1996.
[53] U. Mahlstedt, J. Alt, and M. Heinitz, “Current: A test generation system for IDDQ testing,” Proc. VLSI Test Symp., pp.317–323, April 1995.
[54] Y. Higami, K.K. Saluja, and K. Kinoshita, “Static test compaction for IDDQ testing of sequential circuits,” Dig. Int. Workshop on IDDQ Testing, pp.9–13, Nov. 1998.
[55] 樋上喜信，K.K. Saluja，高松雄三，樹下行三，“順序回路のブリッジ故障に対する IDDQ テストのための静的なテスト系列圧縮法,” 信学論（D-I），vol.J82-D-I, no.7, pp.689–696, July 1999.
[56] T. Lee, I.N. Hajj, E.M. Rundnick, and J.H. Patel, “Genetic-algorithm-based test generation for current testing of bridging faults in CMOS VLSI circuits,” Proc. VLSI Test Symp., pp.456–462, April 1996.
[57] S. Chakravarty and P.J. Thadikaran, “A study of IDDQ subset selection algorithms for bridging faults,” Proc. Int. Test Conf., pp.403–412, Oct. 1994.
[58] S. Chakravarty and P.J. Thadikaran, “Algorithm to select IDDQ measurement points to detect bridging faults,” J. Electron. Test.: Theory Appl., vol.8, pp.275–285, June 1996.
[59] Y. Higami, K.K. Saluja, and K. Kinoshita, “Observation time reduction for IDDQ testing of bridging faults in sequential circuits,” Proc. Asian Test Symp., pp.312–317, Dec. 1998.
[60] Y. Higami, K.K. Saluja, and K. Kinoshita, “Efficient techniques for reducing IDDQ observation time for sequential circuits,” Proc. Int. Conf. on VLSI Design, pp.72–77, Jan. 1999.
[61] H. Kondo and K.T. Cheng, “Driving toward higher IDDQ test quality for sequential circuits: A generalized fault model and its ATPG,” Dig. Int. Conf. on Computer-Aided Design, pp.228–232, Nov. 1996.
[62] B. Koenemann, C. Barnhart, B. Keller, T. Snethen, O. Farnsworth, and D. Wheater, “A smart BIST variant with guaranteed encoding,” Proc. Asian Test Symp., pp.325–330, 2001.
[63] C.V. Krishna, A. Jas, and N.A. Touba, “Test vector encoding using partial LFSR reseeding,” Proc. Int. Test Conf., pp.885–893, 2001.
[64] H. Liang, S. Hellebrand, and H.J. Wunderlich, “Two-dimensional test data compression for scan-based deterministic BIST,” Proc. Int. Test Conf., pp.894–902, 2001.
[65] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois, “Built-in test for circuits with scan based on reseeding of multiple-polynomial linear feedback shift registers,” IEEE Trans. Comput., vol.44, no.2, pp.223–233, Feb. 1995.
[66] P. Wohl, J.A. Waicukauski, S. Patel, and M.B. Amin, “Efficient compression and application of deterministic patterns in a logic BIST architecture,” Proc. Design Automation Conf., pp.566–569, 2003.
[67] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, K.H. Tsai, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eide, and J. Qian, “Embedded deterministic test for low cost manufacturing test,” Proc. Int. Test Conf., pp.301–310, 2002.
[68] 小西秀明，岡埜　靖，山村一之，唐沢直子，板矢剛一，熊谷淳子，江守道明，相京　隆，平出貴久，“ATG および BIST 技術を応用したテストコスト削減の新手法,” 信学技報，FTS2001-78, 2002.
[69] K. Chakrabarty, B.T. Murray, J. Liu, and M. Zhu, “Test width compression for built-in self testing,” Proc. Int. Test Conf., pp.328–337, 1997.
[70] I. Hamzaoglu and J.H. Patel, “Reducing test application time for built-in-self-test test pattern generators,” Proc. VLSI Test Symp., pp.369–375, 2000.
[71] I. Bayraktaroglu and A. Orailoglu, “Test volume and application time reduction through scan chain concealment,” Proc. Design Automation Conf., pp.151–155, 2001.
[72] S.M. Reddy, K. Miyase, S. Kajihara, and I. Pomeranz, “On test data volume reduction for multiple scan chain designs,” Proc. VLSI Test Symp., pp.103–108, 2002.
[73] V. Iyengar, K. Chakrabarty, and B. Murray, “Built-in self testing of sequential circuits using precomputed test sets,” Proc. VLSI Test Symp., pp.418–423, 1998.
[74] A. Jas, J.G.-Dastidar, and N.A. Touba, “Scan vector compression/decompression using statistical coding,” Proc. VLSI Test Symp., pp.114–120, 1999.
[75] M.E. Ng and N.A. Touba, “Test vector compression via statistical coding and dynamic compaction,” Proc. SRTS, pp.348–354, 2000.

[76] H. Ichihara, K. Kinoshita, I. Pomeranz, and S.M. Reddy, “Test transformation to improve compaction by statistical encoding,” Proc. Int. Conf. on VLSI Design, pp.294–299, 2000.

[77] H. Ichihara and T. Inoue, “Generating small test sets for test compression/decompression scheme using statisitical coding,” Proc. Int. Workshop on Electronic Design, Test and Applications, pp.396–400, 2002.

[78] S. Kajihara, K. Taniguchi, I. Pomeranz, and S.M. Reddy, “Test data compression using don’t-care identification and statistical encoding,” Proc. Int. Workshop on Electronic Design, Test and Applications, pp.413–416, 2002.

[79] T. Yamaguchi, M. Tilgner, M. Ishida, and D.S. Ha, “An efficient method for compressing test data,” Proc. Int. Test Conf., pp.79–88, 1997.

[80] M. Ishida, D.S. Ha, and T. Yamaguchi, “COMPACT: A hybrid method for compressing test data,” Proc. VLSI Test Symp., pp.62–69, 1998.

[81] A. Jas and N.A. Touba, “Test vector decompression via cyclical scan chains and its application to testing core-based designs,” Proc. Int. Test Conf., pp.458–464, 1998.

[82] A. Chandra and K. Chakrabarty, “System-on-a-chip test data compression and decompression architectures based on Golmb codes,” IEEE Trans. Comput., vol.20, no.3, pp.355–368, March 2001.

[83] A. Chandra and K. Chakrabarty, “Frequency-directed run-length (FDR) codes with application to system-on-a-chip test data compression,” Proc. VLSI Test Symp., pp.42–47, 2001.

[84] 小原敏敬，新谷道広，市原英行，井上智生，田村秋雄，“ハフマン符号に基づくテスト実行のためのテスト圧縮について，” 信学技報，DC2002-90, 2003.

[85] A. Jas and N.A. Touba, “Using an embedded processor for efficient deterministic testing of system-on-a-chip,” Proc. Int. Conf. on Computer Design, pp.418–423, 1999.

[86] A. El-Maleh, S. al Zahir, and E. Khan, “A geometric-primitives-based compression scheme for testing system-on-a-chip,” Proc. VLSI Test Symp., pp.54–59, 2001.

[87] 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.

[88] 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.

[89] H. Fujiwara and A. Yamamoto, “Parity-scan design to reduce the cost of test application,” Proc. Int’l Test Conf., pp.283–292, 1992.

[90] H. Fujiwara and A. Yamamoto, “Parity-scan design to reduce the cost of test application,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.12, no.10, pp.1604–1611, 1993.

[91] 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.

[92] Y. Higami, S. Kajihara, and K. Kinoshita, “A reduced scan shift method for sequential circuit testing,” IEICE Trans. Fundamentals, vol.E77-A, no.12, pp.2010–2016, Dec. 1994.

[93] 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.

[94] 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.

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

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

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

[98] 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.

[99] 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.

[100] 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，2003.

[101] Y. Higami and K. Kinoshita, “Design of partially parallel scan chain,” Proc. European Design and Test Conf., p.626, March 1997.

[102] 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.

[103] S. Narayanan, C. Njinda, and M. Breuer, “Optimal sequencing of scan registers,” Proc. Int’l Test Conf, pp.293–302, Sept. 1992.

（平成 15 年 4 月 3 日受付，8 月 12 日再受付）

樋上　喜信　（正員）

平 8 阪大大学院工学研究科応用物理学専攻博士後期課程了．同年日本学術振興会特別研究員採用．平 10 より愛媛大学工学部助手．現在同学部講師．論理回路に対するテスト生成及びテスト容易化設計に関する研究に従事．情報処理学会，IEEE 各会員．博士（工学）．

梶原　誠司　（正員）

昭 62 広島大・総合科学・総合科学卒，平 4 阪大大学院工学研究科博士後期課程了．博士（工学）．同大・工・応用物理助手を経て，平 8 九州工大・情報工学・電子情報助教授．平 15 より，同大教授．この間，平 9〜11 阪大大学院工学研究科助教授（併任）．VLSI のテスト生成，テスト容易化設計などの研究に従事．平 8 年度本会学術奨励賞，平 14 情報処理学会山下記念研究賞など受賞．情報処理学会，IEEE 各会員．

市原　英行　（正員）

平 7 阪大・工・応用物理卒．平 9 同大大学院工学研究科応用物理学専攻博士前期課程了．平 11 同専攻博士後期課程了，博士（工学）．同年広島市立大学情報科学部情報機械システム工学科助手，現在に至る．VLSI 回路のテスト生成，テストデータ圧縮，テスト容易化設計などの研究に従事．IEEE 会員．

高松　雄三　（正員）

昭 41 愛媛大・工・電気卒．佐賀大理工学部電子工学科助教授を経て，昭 62 年 10 月より愛媛大学工学部情報工学科教授，現在に至る．論理回路のテスト生成法及び故障診断法などに関する研究に従事．工博．平 6 IEEE 第 3 回アジアテストシンポジウムプログラム委員長，平 9 IEEE 第 6 回同シンポジウム実行委員長など．著書「新版・論理設計入門」（共著，日新出版）など．情報処理学会会員，IEEE Senior Member．