組合せ回路および順序回路に対する検出・非検出情報に基づく診断用テスト圧縮法

樋上喜信[†] ケーワル K. サルージャ[††] 高橋寛[†]
小林真也[†] 高松雄三[†]

近年，論理回路のテストや故障診断におけるコスト削減が重要になってきている．テストや故障診断のコストは，印加されるテストベクトル数に依存するため，コスト削減のためにはテストベクトルを削減することが重要である．本論文では，組合せ回路および順序回路に対して，故障診断のためのテストベクトル数削減法（テスト圧縮法）を提案する．提案するテスト圧縮法では，与えられたテスト集合またはテスト系列に対して，区別される故障ペア数を減少させることなく，テストベクトル数を削減する．故障ペア数は故障数の 2 乗に比例するため，大規模回路においてそれは膨大な数となる．そこで提案法では発見的手法を用いて，一度に取り扱う故障ペア数を減少させることによって，大規模回路においてもテストベクトル削減を可能にする．なお提案法では，検出・非検出情報に基づく故障診断を仮定する．これは，故障検出の有無だけの情報を用い，故障影響が観測される外部出力情報を用いない故障診断のことである．提案法の有効性は，ISCAS ベンチマーク回路に対する実験の結果によって示される．

Compaction of Pass/Fail-based Diagnostic Test Vectors for Combinational and Sequential Circuits

Yoshinobu Higami,[†] Kewal K. Saluja,[††] Hiroshi Takahashi,[†]
Shin-ya Kobayashi[†] and Yuzo Takamatsu[†]

Recently, it is getting more important to reduce the cost of test and fault diagnosis. Since the cost of test and fault diagnosis depends on the number of test vectors, test vectors must be compacted. This paper presents methods for compacting of pass/fail-based diagnostic test sets or test sequences for combinational and sequential circuits. The pass/fail-based diagnosis uses only pass/fail information of test vectors but not information on location of primary outputs where faulty effects are observed. The proposed methods reduce the number of test vectors while maintaining the original diagnostic capability. In order to compact diagnostic test vectors, we must take care of a large number of fault pairs, which is the square number of faults. The proposed methods introduce heuristics to reduce the number of fault pairs that are handled at one time. The effectiveness of the proposed methods are shown by experimental results for ISCAS benchmark circuits.

1. はじめに

最近の LSI のテストにおいては，テストコストの増大が深刻な問題となってきている．大規模 LSI では，非常に多くのテストベクトルを印加する必要があり，その結果，テスト印加時間やテスタに必要なメモリ量が増大し，最終的にはテストコストの増大を招いている．これまでに，テスト印加時間の削減およびテストデータ量削減のための手法が数多く提案されており，これらはテスト圧縮法とよばれている[1)]．文献 9) では，テスト圧縮の故障診断への影響を実験的に調査し，テスト圧縮を行っても診断能力が大きく失われることがないことを示している．これまで，故障診断用のテスト集合やテスト系列を圧縮する研究はほとんど報告されていないが，故障診断においてもテスト圧縮を行うことは重要であると考えられる．テストベクトル数が減少することによって，実際に故障診断に要する時間の減少，テスタメモリ量の減少などの効果が期待される．たとえば組み込み自己テスト（BIST）において故障が検出された回路に対して故障診断を行う場合，BIST の出力は通常圧縮されたシグネチャであり，それをもとに故障診断を行うことは一般的に困難

† 愛媛大学
Ehime University
†† ウィスコンシン大学
University of Wisconsin - Madison

である．したがってテスタを用いてもう一度テストベクトルを印加し，各テストベクトルに対する出力応答を得る必要がある．このような場合，BIST で用いられたテストベクトルをすべて用いることは実用的ではなく，少ないテストベクトルを抽出することが有効である．また故障診断法の 1 つである故障辞書法においても，テストベクトルを削減することによって必要となる故障辞書の大きさが減少する．

本論文では，組合せ回路および順序回路に対する故障診断用テスト集合およびテスト系列に対するテスト圧縮法を提案する．ここでは，BIST を考慮した検出・非検出情報による故障診断を仮定する[2]．すなわち，被検査回路に印加した各テストベクトルに対して，故障検出の有無の情報のみを用い，故障が検出された外部出力（またはスキャンフリップフロップ）の位置情報を用いずに故障診断を行う☆．

提案する組合せ回路に対する診断用テスト圧縮法では，故障区別表を用いる．故障区別表は，各故障ペアが各テストベクトルで区別されるかどうかを表している．もし，すべての故障ペアとすべてのテストベクトルに対する故障区別表（完全故障区別表とよぶ）が与えられたならば，テスト圧縮の問題は，最小被覆問題となり近似的な最適解を容易に得ることができる．しかしながら，一般的に故障ペア数は膨大であるため，完全故障区別表を利用することは困難である．そこで提案法では，一部の故障ペアを選択し，それらに対する故障区別表を作成し，選択した故障ペアのすべてを区別するテストベクトルを選択する．さらに別の故障ペアを選択し，同様の計算を繰り返すことによってテスト圧縮を行う．

診断用テスト生成[3),5)]や故障シミュレーション[8),10)]の場合と同様，順序回路に対する診断用テスト圧縮は，組合せ回路に対するそれより困難である．提案法では，逆順ベクトル回復法（Reverse Order Restoration）とよばれる手法[4)]を応用する．逆順ベクトル回復法は故障検出を目的とした順序回路に対するテスト圧縮法で，高いテスト圧縮能力を有している．故障診断を目的として逆順ベクトル回復法を適用する場合，各故障ペアが区別される時刻の情報が必要であり，故障ペア数が多い回路に対してはそのまま適用することは難しい．そこで提案法では発見的手法によって，逆順ベクトル回復法で一度に扱う故障ペア数を減少させ，大規模回路でも適用可能にした．提案法の有効性は ISCAS ベンチマーク回路に対する実験によって確認する．

本論文の以下の構成は次のとおりである．2 章では準備としていくつかの定義を行う．3 章では組合せ回路に対する診断用テスト圧縮法を，4 章では順序回路に対する診断用テスト圧縮法を説明する．最後に 5 章で本論文のまとめを述べる．

2. 準　　備

本論文で考える診断用テスト圧縮の問題は，与えられたテスト集合またはテスト系列に対して，故障診断能力を低下させずにテストベクトル数をできるだけ減少させることである．

提案法を説明するため，いくつかの定義を行う．

［定義 1］故障 f_1 が存在する回路にテストベクトル t を印加したときと，故障 f_2 が存在する回路にテストベクトル t を印加したときにおいて，故障 f_1 が検出されかつ故障 f_2 が検出されない，または故障 f_2 が検出されかつ故障 f_1 が検出されないならば，故障 f_1 と f_2 はテストベクトル t によって**区別される**という．

［定義 2］各故障が各テストベクトルに対して，検出されるかどうかを示した表を故障検出表とよぶ．また各故障ペアが各テストベクトルに対して，区別されるかどうかを示した表を故障区別表とよぶ．

［定義 3］状態が s である順序回路に対して，テストベクトル $v_1, v_2, \ldots, v_k$ $(k \geq 1)$ を印加したとき，s がどのような状態であっても必ず特定の状態 s_i に遷移するならば，$v_1, v_2, \ldots, v_k$ を初期化ベクトルとよぶ．

本論文では，元のテスト集合またはテスト系列で検出される故障のみを取り扱い，検出されない故障は考慮しない．これは，故障診断は故障が存在すると識別された回路に対してのみ行うため，検出されない故障が存在する回路に対して故障診断することはないためである．

3. 組合せ回路に対するテスト圧縮法

3.1 基本的な考え方

もしすべての故障ペアとすべてのテストベクトルに対する完全故障区別表が与えられたならば，テスト圧縮の問題は最小被覆問題となり，近似的最適解を容易に得ることができる．しかしながら一般に故障ペア数は膨大であるため，故障区別表を計算機上に記憶することが困難である．そこで，一部の故障ペアに対する故障区別表を作成し，それらの故障ペアを区別するような近似的最小数のテストベクトルを選択する．さらに別の故障ペアを選択し，同様にしてテストベクトル

☆ 本研究は BIST を前提としているが，提案法は通常のテスタ環境の診断にも適用可能であり，さらに回路内部に観測点を設けるような診断容易化設計法[7)]にも拡張して適用することも可能である．

を選択する．このような計算をすべての故障ペアが区別されるまで繰り返す．以下の定理は，テスト集合中の 1 つのテストベクトルでしか検出されない故障と，それを検出するテストベクトルについて述べており，そのようなテストベクトルは 1 つを除きすべてを，圧縮したテスト集合に含むべきであることを示している．

［定理 1］m 個のテストベクトルからなるテスト集合 V_0 が与えられとき，V_0 中の 1 つのテストベクトルでしか検出されない故障の集合を Fd_1 とし，V_0 中で故障 $^{\forall}f \in Fd_1$ を検出するテストベクトルの集合を V_1 とする．また Fd_1 に含まれる故障からなる故障ペアのうち，テストベクトル $^{\forall}v \in V_0$ によって区別される故障ペアの集合を FP とする．もし FP 中の区別可能な故障ペアをすべて区別するテストベクトルを V_0 から選択するならば，少なくとも V_1 中の $m-1$ 個のテストベクトルが必要である．

（証明）テストベクトル v_1 と v_2 を V_1 中のテストベクトル，f_1 と f_2 をそれぞれ v_1 と v_2 に検出され，かつ集合 Fd_1 に含まれる故障とする．テストベクトル v_1 と v_2 を V_1 から削除した集合を V_1' としたとき，f_1 と f_2 からなる故障ペアは，V_1' 中のどのテストベクトルによっても区別されない．なぜなら f_1 と f_2 は，v_1 または v_2 以外のテストベクトルには検出されないからである．したがって，テスト集合 V_1 から 2 つのテストベクトルを削除した場合，区別されない故障ペアが生じるため，すべての故障ペアを区別するためには少なくとも V_1 中の $m-1$ 個のテストベクトルが必要である．

3.2 提 案 手 法

本研究で考える組合せ回路に対するテスト圧縮問題は，次のように定式化される．

組合せ回路に対する診断用テスト圧縮の問題

入力：テスト集合（V）

出力：V の部分集合（V_s）

目標：V_s に含まれるテストベクトル数をできるだけ少なくする．

条件：V で検出される故障からなるペアのうち V で区別されるペアをすべて区別する．

上記の問題を解決するための提案法（DCOMP-C とよぶ）を説明する．DCOMP-C では定理 1 に従い，まず 1 つのテストベクトルでしか検出されない故障を検出するテストベクトルを選択する．次に，得られたテストベクトルを用いて故障シミュレーションを行い，区別されない故障ペアの中から一定数の故障ペアをランダムに選択する．選択した故障ペアに対して故障区別表を作成し，それらの故障ペアを区別するテストベクトルを選択する．故障ペア数を一定数に制限することによって，故障区別表の作成を可能にする．このような故障ペアの選択とそれらを区別するテストベクトルの選択を繰り返すことによって，元々区別されていた故障ペアをすべて区別するようなテストベクトルを選択する．DCOMP-C の詳細を以下に示す．

組合せ回路に対する診断用テスト圧縮法 DCOMP-C

/* V_0：与えられたテスト集合 */

1) $V_s = \phi$.

2) V_0 を用いて故障シミュレーションを行い，V_0 中の 1 つのテストベクトルでしか検出されない故障の集合を求め，それを Fd_1 とする．

3) Fd_1 に含まれる故障を検出するテストベクトルを集め，それらの集合を V_s とする．またそれらのテストベクトルを V_0 から削除する．

4) V_s を用いて故障シミュレーションを行い，$^{\forall}v \in V_s$ で区別されない故障ペアの中から n_p 個の故障ペアをランダムに選択する．選択した故障ペアの集合を P とする．

5) V_0 と P に対して故障区別表を作成する．

6) P 中のどの故障ペアもテストベクトル $^{\forall}v \in V_0$ によって区別されないならば，8) へ進む．

7) 故障区別表を用いて最小被覆問題を解くことによって，P 中の区別可能なすべての故障ペアを区別するテストベクトルを V_0 から選択する．選択したテストベクトルを V_0 から削除し，V_s へ加える．4) へ戻る．

8) V_s 中のどのテストベクトルによっても区別されない故障ペアをすべて求め，その集合を P とする．

9) V_0 と P に対して故障区別表を作成する．

10) 最小被覆問題を解くことによって，P 中の区別可能なすべての故障ペアを区別するテストベクトルを V_0 から選択する．選択したテストベクトルを V_s へ加える（V_s が最終的に得られた圧縮されたテスト集合を表す）．

上記 DCOMP-C の手順について説明する．まず 2), 3) の手順において，1 つのテストベクトルでしか検出されない故障を求め，次にそれらの故障を検出するテストベクトルを求め，集合 V_s とする．定理 1 により，これらのテストベクトルが必要であることが示されている（正確には，1 つを除いたすべてのテストベクトルで十分であるが，ここではすべてを選択する）．4), 5) の手順では，その時点で得られているテスト集合 V_s によって区別されない故障ペアの中から，n_p 個の故障ペアをランダムに選択する．n_p はあらかじめ決められた定数とする．次に n_p 個の故障ペアとテスト

表 1 故障検出表
Table 1 Fault detection table.

	v_1	v_2	v_3	v_4	v_5
f_1	d				
f_2		d	d	d	
f_3			d	d	
f_4			d	d	
f_5				d	d
f_6				d	d
f_7			d	d	d

d：検出を表す

集合 V_0 に対して，故障区別表を作成する．テスト集合 V_0 には，与えられた元のテスト集合から V_s に選択されたテストベクトルを除いたもののみが含まれている．故障区別表が作成された後は，最小被覆問題を解くことによって近似的最小数のテストベクトルを選択し，それを V_s に加える．このような処理は，選択した n_p 個の故障ペアすべてが V_0 によって区別されなくなるまで続けられる．十分多くのテストベクトルを選択した後には，選択した n_p 個の故障ペアのすべてが V_0 によって区別されなくなると考えられる．そのような場合，8) の手順において，残っている V_0 内のテストベクトルと，V_s で区別されない故障ペアすべてに対して故障区別表を作成し，近似的最小数のテストベクトルを V_0 から選択し，それを V_s に加える．最終的に得られた V_s が圧縮されたテスト集合であり，元の V_0 で区別された故障ペアのすべてが V_s で区別されることが保証される．

例 1：テスト集合 $V_0 = \{v_1, v_2, v_3, v_4, v_5\}$ と，テストベクトル $v \in V_0$ によって検出される故障 $f_1, f_2, \ldots, f_7$ を考える．V_0 に対して $f_1, f_2, \ldots, f_7$ がどのテストベクトルで検出されるかについては，**表 1** のようであったとする（ただしこのような故障検出表を明に用いるわけではない）．このような例に対して DCOMP-C を適用した場合，次のように計算が実行される．2) では 1 つのテストベクトルでしか検出されない故障が集められ，この例では $Fd_1 = \{f_1\}$ が得られる．次に 3) では Fd_1 に含まれる故障を検出するテストベクトルが集められ，$V_s = \{v_1\}$ が得られる．4) では v_1 を用いて故障シミュレーションを行い，v_1 で区別されない故障ペアの中から n_p 個の故障ペアを選択する．この例においては，$n_p = 2$ と仮定し，$P = \{< f_2, f_3 >, < f_4, f_5 >\}$ が得られたとする．ここで $< f_i, f_j >$ は故障 f_i と f_j からなる故障ペアを表す．5) では $V_0 = \{v_2, v_3, v_4, v_5\}$ を用いて故障シミュレーションを行い，**表 2** に示されるような故障区別表を作成し，7) において最小被覆問題を解くことによって，P に含まれる故障ペアを区別する近似的最小数のテストベクトルを得る．ここでは v_2 と v_3 が選択されたとする．次に処理は 4) に戻り，故障ペアの集合 $P = \{< f_3, f_4 >, < f_5, f_6 >\}$ が得られたとする．6) において，故障ペア $< f_3, f_4 >$，$< f_5, f_6 >$ の両方が $V_0 = \{v_4, v_5\}$ によって区別されないので，処理は 8) に進む．ここでは，V_s によって区別されない故障ペアのすべてが集められ，$P = \{< f_3, f_4 >, < f_3, f_7 >, < f_4, f_7 >, < f_5, f_6 >\}$ が得られる．9) において故障区別表を作成し，10) において故障ペア $< f_3, f_7 >$，$< f_4, f_7 >$ を区別するテストベクトル v_5 が選択される．最終的には圧縮したテスト集合として，$V_s = \{v_1, v_2, v_3, v_5\}$ が得られる．

表 2 故障区別表
Table 2 Fault distinguishing table.

	v_1	v_2	v_3	v_4	v_5
$< f_2, f_3 >$		D			
$< f_4, f_5 >$			D		D

D：区別を表す

3.3 実験結果

DCOMP-C を C 言語を用いてプログラム化し，ISCAS'85 組合せ回路と ISCAS'89 順序回路の組合せ回路部分を用いて実験を行った．実験では Pentium IV 2.6 GHz 計算機を用い，1,024 のランダムテストを元のテスト集合とした．**表 3** に結果を示す表中の各列には左から，回路名（ckt），故障検出率（cov），検出された故障からなる故障ペア数（pair），そのうち区別されない故障ペア数（undis），圧縮されたテストベクトル数（vect），元のテストベクトル数に対する削減されたテストベクトル数の割合（%）をそれぞれ示す．**表 4** には，計算時間を示す．表 3，表 4 とも，n_p = 100,000，n_p = 10,000，n_p = 1,000 の各列には，変数 n_p をそれぞれ 100,000，10,000，1,000 に設定した場合の結果を示す．実験の結果，n_p = 100,000 としたとき 1,024 テストベクトルを 37〜509 テストベクトルにまで圧縮することができた．さらに s35932 では 6 億以上の故障ペアを対象にテスト圧縮を実現できた．n_p を変化させたときの結果を比較すると，多くの回路において n_p = 100,000 の場合に短い計算時間で少ないテストベクトル数が得られた．

4. 順序回路に対するテスト圧縮法

4.1 逆順ベクトル回復法

順序回路に対するテスト圧縮は，組合せ回路に対する場合より困難である．順序回路に対するテスト系列において，あるテストベクトルを削除した場合には，そのテストベクトル以降における状態遷移が変化し，その結果元々検出されていた故障が検出されなくなる

表 3 DCOMP-C を適用した実験結果
Table 3 Experimental results by DCOMP-C.

				$n_p = 100{,}000$		$n_p = 10{,}000$		$n_p = 1{,}000$	
ckt	cov	pair	undis	vect	%	vect	%	vect	%
c432	97.52	$1.30 * 10^5$	93	68	93.4	72	93.0	71	93.1
c880	97.52	$4.45 * 10^5$	104	63	93.8	65	93.7	70	93.2
c1355	98.57	$1.25 * 10^6$	878	88	91.4	88	91.4	89	91.3
c1908	94.12	$1.84 * 10^6$	1,208	139	86.4	144	85.9	144	85.9
c2670	84.40	$3.08 * 10^6$	1,838	79	92.3	76	92.6	79	92.3
c3540	94.49	$5.95 * 10^6$	1,585	205	80.0	199	80.6	207	79.8
c5315	98.83	$1.57 * 10^7$	1,579	188	81.6	184	82.0	199	80.6
c6288	99.56	$2.97 * 10^7$	4,491	37	96.4	37	96.4	38	96.3
c7552	91.97	$2.76 * 10^7$	4,438	198	80.7	193	81.2	208	79.7
cs5378	94.55	$9.47 * 10^6$	1,802	231	77.4	239	76.7	245	76.1
cs9234	73.86	$1.31 * 10^7$	6,798	246	76.0	256	75.0	261	74.5
cs15850	85.05	$4.97 * 10^7$	9,368	261	74.5	265	74.1	279	72.8
cs35932	89.81	$6.16 * 10^8$	16,655	91	91.1	94	90.8	96	90.6
cs38417	86.58	$3.64 * 10^8$	12,480	436	57.4	445	56.5	453	55.8
cs38584	90.60	$5.41 * 10^8$	31,607	509	50.3	529	48.3	537	47.6

表 4 DCOMP-C を適用した実験結果（計算時間 (s)）
Table 4 Experimental results by DCOMP-C (runtime (s)).

ckt	$n_p = 100{,}000$	$n_p = 10{,}000$	$n_p = 1{,}000$
c432	0.1	0.1	0.1
c880	0.2	0.3	0.3
c1355	0.8	1.1	1.3
c1908	2.1	2.5	2.8
c2670	2.8	2.7	3.9
c3540	10.6	9.9	16.6
c5315	15.4	14.5	25.1
c6288	1,659	2,098	3,560
c7552	33.8	32.3	110.5
cs5378	12.2	15.5	21.0
cs9234	36.1	41.7	100.6
cs15850	141.8	171.9	389.4
cs35932	962.2	1,244	2,023
cs38417	1,848	2,572	7,021
cs38584	2,069	7,328	10,030

可能性がある．したがって，元のテスト系列と同じ故障検出率や同じ故障診断分解能を維持するためには，故障シミュレーションなどを行って，それを保証する必要がある．これまで故障検出を目的とした順序回路に対する有効なテスト圧縮法として，逆順ベクトル回復法が提案されている[4]．逆順ベクトル回復法では，まず初期化ベクトルを除くすべてのテストベクトルを削除する．次に元のテスト系列中の最も遅い時刻で検出される故障を求め，それを検出するように選択したテストベクトルをテスト系列に加える．さらにまた別の故障に対しても同様に，テストベクトルを順にテスト系列に加える．このような処理をすべての故障が検出されるまで繰り返す．以下に例を用いて，逆順ベクトル回復法を説明する．

表 5 検出故障表の例
Table 5 Example of a fault detection table.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
f_1			d				
f_2				d			
f_3							d

d：検出を表す

例 2：7 つのテストベクトルからなるテスト系列 T_0 が与えられ，T_0 は故障 f_1，f_2，f_3 を検出すると仮定する．**表 5** は T_0 に対する故障検出表を表す．逆順ベクトル回復法では，まず初期化ベクトルを除くすべてのテストベクトルを削除する．ここでは，v_1，v_2 を初期化ベクトルとし，まずそれらでテスト系列 T_c を構成する．次に元のテスト系列中で最も遅い時刻で検出される故障を求める．この例では f_3 が該当し，それを検出するテストベクトル v_7 を T_c に加える．故障シミュレーションを行い，$T_c = v_1 - v_2 - v_7$ が故障 f_3 を検出するかどうか調べる．もしそれを検出したならば，T_c によって検出される故障を故障リストから除く．もし故障 f_3 を検出しないならば，テストベクトル v_6 を T_c に加え，$T_c = v_1 - v_2 - v_6 - v_7$ が故障 f_3 を検出するかどうか調べる．T_c が f_3 を検出したならば，故障 f_2 を検出するようにテストベクトル v_4 を T_c に追加する．このような処理をすべての故障が検出されるまで繰り返す．

4.2 基本的なテスト圧縮法

本研究で考える順序回路に対するテスト圧縮問題は，次のように定式化される．

順序回路に対する診断用テスト圧縮の問題
入力：テスト系列（T_0）

出力：T_0 の一部のテストベクトルからなるテスト系列（T_c）
目標：T_c に含まれるテストベクトル数をできるだけ少なくする．
条件：T_0 で検出される故障からなるペアのうち T_0 で区別されるペアをすべて区別する．

上記の問題を解決する提案手法として，逆順ベクトル回復法に基づく診断用テスト圧縮法（DCOMP-S とよぶ）を説明する．逆順ベクトル回復法と同様，DCOMP-S でもまず初期化ベクトルを除くすべてのテストベクトルを削除する．次に元のテスト系列中で最も遅い時刻で区別される故障ペアを抽出し，それらを区別するようなテストベクトルをテスト系列に追加する．この処理をすべての故障ペアが区別されるまで繰り返す．提案法の詳細を以下に示す．

順序回路に対する診断用テスト圧縮法：DCOMP-S

/* T_0：与えられたテスト系列 */
/* v_i：T_0 中の i 番目のテストベクトル */
/* F_0：T_0 によって検出される故障の集合 */
/* FP_0：故障 $f \in F_0$ から構成された故障ペアで T_0 によって区別される故障ペアの集合（各故障ペアが区別される時刻の情報を含む）*/
1) $T_c = \phi$，$FP_{trg} = \phi$.
2) v_1 から v_{init} のテストベクトルを初期化ベクトルとし，それらを T_c に加える．
3) T_c によって区別される故障ペアを FP_0 から削除する．
4) FP_0 中の故障ペアが T_0 によって区別される時刻の中で最も遅い時刻を求め，それを t とする．
5) テストベクトル v_t によって区別される故障ペアをすべて FP_{trg} に加える．
6) テストベクトル v_t をテスト系列 T_c に加える．
7) FP_{trg} 中の故障ペアがすべて T_c によって区別されるか調べる．
8) もし 1 つでも区別されない故障ペアが存在したならば，$t = t - 1$ とし，5) に戻る．
9) T_c によって区別される故障ペアを FP_0 から削除し，$FP_{trg} = \phi$ とする．
10) もし $FP_0 = \phi$ なら，終了する．そうでないなら，$t = t - 1$ とし，5) へ戻る．

上記の手順について説明する．DCOMP-S では，まず初期化ベクトルのみをテスト系列 T_c に加える．初期化ベクトルとしては，T_0 中の 1 番目から $init$ 番目までのテストベクトルを抽出し T_c に加える．$init$ は小さな値とする．3) では故障シミュレーションを行い，T_c で区別される故障ペアを FP_0 から除く．次に 4) において，各故障ペアが T_0 において区別される時刻を調べ，その中で最も遅い時刻を求め，その時刻を t とする．故障ペアが複数の時刻で区別される場合には，その中で最も遅い時刻のみを考慮する．5) では，テスト系列 T_0 において時刻 t で区別される故障ペアを抽出し，それらを集合 FP_{trg} に加える．6) では，時刻 t のテストベクトル v_t を T_c に加える．もし FP_{trg} が時刻 t で区別される故障ペアのみを含んでいるならば，v_t を T_c の 1 番最後に加える．もしそうでない場合には，T_c に必ず v_{t+1} が含まれているので，v_t を v_{t+1} の前に挿入する．7) では，FP_{trg} の故障ペアが T_c によってすべて区別されるかどうかを調べる．もし 1 つでも区別されない故障ペアがあったならば，時刻 t を 1 減らし，5) へ戻る．もし FP_{trg} のすべての故障ペアが区別されたならば，さらに T_c で区別されるその他の故障ペアを求め，FP_0 から削除する．

4.3 DCOMP-S を適用した実験結果

DCOMP-S を C 言語を用いてプログラム化し，ISCAS'89 ベンチマーク回路に対して実験を行った結果を示す．用いた計算機は Pentium IV 2.6 GHz であり，与えられるテスト系列 1,024 個のランダムベクトルからなるテスト系列を用いた．**表 6** の各列には左から順に，回路名（ckt），故障検出率（cov(%)），テスト系列長（len），圧縮されたテスト系列長（comp），元の系列長に対する削減されたテストベクトル数の割合（%），元のテスト系列で検出された故障からなる故障ペア数（pair），そのうち区別されない故障ペア数（undis），計算時間（cpu(s)）を表す．変数 $init$ は，元の系列長の 2%とした．表 6 の結果より，すべての回路においてテスト系列を圧縮することができた．しかしながら，ランダムベクトルによって低い故障検出率しか達成されていない回路が存在するため，縮退故障用テスト生成ソフトウエア HITEC[6] で生成された縮退故障用テスト系列を用いた実験を行った．**表 7** に実験の結果を示す．表 7 の各列は，表 6 の各列と同様のデータを示している．実験の結果，s526 回路に対して最も多くのテストベクトルを削減することができ，約 72%のテストベクトルを削減することができた．これらの実験で用いた回路は比較的小規模の回路であり，大規模回路に対しては，故障ペア数が膨大であったため，DCOMP-S を適用することができなかった．次の節で，そのような大規模回路に適用可能な手法について説明する．

4.4 大規模回路に対する手法

DCOMP-S はすべての故障ペアに対して，区別さ

表 6 ランダムベクトルに対して DCOMP-S 適用した実験結果
Table 6 Experimental results by DCOMP-S with random vectors.

ckt	cov.	len	comp	%	pair	undis	cpu(s)
s344	93.1	1,024	104	89.8	47,895	359	0.4
s349	92.7	1,024	100	90.2	49,770	379	0.4
s382	12.3	1,024	26	97.5	1,176	253	0.0
s386	64.3	1,024	86	91.6	30,381	290	0.3
s400	12.0	1,024	25	97.6	1,275	300	0.0
s444	11.2	1,024	25	97.6	1,378	309	0.0
s526	9.4	1,024	23	97.8	1,326	760	0.0
s641	83.2	1,024	119	88.4	74,691	133	0.6
s713	79.4	1,024	129	87.4	106,030	384	0.8
s820	33.3	1,024	99	90.3	39,903	371	0.5
s832	32.4	1,024	98	90.4	39,621	371	0.6
s1196	83.8	1,024	232	77.3	541,320	371	12.5
s1238	79.0	1,024	239	76.7	572,985	481	14.5
s1423	36.8	1,024	187	81.7	155,403	2,049	6.0
s1488	55.3	1,024	106	89.6	336,610	1,667	3.0
s1494	54.2	1,024	106	89.6	332,520	1,637	3.1
s5378	64.5	1,024	847	17.3	4,241,328	3,193	942.1

表 7 HITEC テスト系列に対して DCOMP-S 適用した実験結果
Table 7 Experimental results by DCOMP-S with HITEC sequences.

ckt	cov.	len	comp	%	pair	undis	cpu(s)
s344	95.8	127	82	35.4	50,721	367	0.3
s349	95.3	134	85	36.6	52,650	402	0.3
s382	78.2	2,074	846	59.2	48,516	1,806	16.4
s386	81.8	286	164	42.7	49,141	106	0.7
s400	82.6	2,214	845	61.8	61,075	2,343	24.9
s444	82.1	2,240	956	57.3	75,466	2,994	21.3
s526	65.1	2,258	631	72.1	64,980	6,617	19.4
s641	86.5	209	123	41.1	80,601	129	0.7
s713	81.9	173	113	34.7	113,050	383	0.8
s820	95.7	1,115	745	33.2	330,078	284	33.9
s832	93.9	1,137	761	33.1	333,336	270	42.6
s1196	99.8	435	295	32.2	766,941	167	18.2
s1238	94.7	475	316	33.5	822,403	191	22.9
s1423	47.7	150	134	10.7	261,003	3,823	4.9
s1488	97.2	1,170	787	32.7	1,041,846	355	75.7
s1494	96.5	1,245	772	38.0	1,054,878	411	106.2
s5378	70.4	912	594	38.0	5,064,153	5,912	1,017.4

れる時刻の情報を保持する必要があったため，小規模回路にしか適用することができなかった．そこで区別される時刻の情報を記憶する故障ペア数を減少させることによって，大規模回路にも適用可能な手法（DCOMP-LS とよぶ）を提案する．DCOMP-LS の手順は DCOMP-S とほぼ同様であるが，一番最初に選択するテストベクトルと，逆順ベクトル回復法で対象とする故障ペアの求め方が異なる．また DCOMP-S では，T_0 で区別されるすべての故障ペアについて，それらが区別される時刻の情報が与えられると仮定したが，DCOMP-LS では，そのような情報を必要としない点も異なる．DCOMP-LS ではまず，元のテスト系列（T_0 とする）の先頭から L 個のテストベクトルを選択し，圧縮系列（T_c とする）に加える．次に T_0 で区別され T_c で区別されない故障ペア（FP_{trg} とする）を故障シミュレーションによって求める．ここで L の値を十分大きな数とすることによって，FP_{trg} に含まれる故障ペア数を計算機上に保存可能な程度に小さくすることができる．FP_{trg} が得られた後は，DCOMP-S と同様に，逆順ベクトル回復法によってテスト圧縮を行う．DCOMP-LS の手順を以下に示す．

大規模順序回路に対する診断用テスト圧縮法：DCOMP-LS

/* T_0：与えられたテスト系列 */
/* v_i：T_0 中の i 番目のテストベクトル */
1) $T_c = \phi$.
2) v_1 から v_L のテストベクトルを T_c に加える．
3) T_0 と T_c を用いて故障シミュレーションを行い，T_0 で区別され T_c で区別されない故障ペアを集め，集合 FP_0 とする．
4) DCOMP-S の 4)〜10) と同一の手順．

表 8 HITEC 系列に対する DCOMP-S と DCOMP-LS の結果の比較
Table 8 Comparison of the results between DCOMP-S and DCOMP-LS with HITEC.

	compacted vectors				cpu(s)	
ckt	S	%	LS	%	S	LS
s344	82	35.4	98	22.8	0.3	0.1
s349	85	36.6	111	17.2	0.3	0.1
s382	846	59.2	1,580	23.8	16.4	18.5
s386	164	42.7	233	18.5	0.7	0.6
s400	845	61.8	1,865	15.8	24.9	42.4
s444	956	57.3	1,648	26.4	21.3	34.4
s526	631	72.1	1,468	35.0	19.4	19.6
s641	123	41.1	177	15.3	0.7	0.4
s713	113	34.7	154	11.0	0.8	0.4
s820	745	33.2	940	15.7	33.9	19.1
s832	761	33.1	998	12.2	42.6	31.5
s1196	295	32.2	366	15.9	18.2	12.5
s1238	316	33.5	398	16.2	22.9	17.7
s1423	134	10.7	145	3.3	4.9	2.6
s1488	787	32.7	992	15.2	75.7	35.2
s1494	772	38.0	1,074	13.7	106.2	47.1
s5378	594	38.0	757	17.0	1,017.4	283.3

表 9 s35932 回路に対する DCOMP-LS 適用の結果
Table 9 Experimental results by DCOMP-LS for s35932.

seq	L(%)	cov(%)	len	comp	%	pair	undis	cpu(s)
H	90	89.28	496	496	0	$6.090 * 10^8$	$6.641 * 10^6$	4,170
H	80	89.28	496	496	0	$6.090 * 10^8$	$6.641 * 10^6$	14,920
H+R	90	89.34	546	535	2.0	$6.098 * 10^8$	$6.641 * 10^6$	2,360
H+R	80	89.34	546	543	0.5	$6.098 * 10^8$	$6.641 * 10^6$	24,163
R	90	70.08	1,024	928	9.4	$3.753 * 10^8$	$1.121 * 10^8$	3,853
R	80	70.08	1,024	863	15.7	$3.753 * 10^8$	$1.121 * 10^8$	7,049

4.5 DCOMP-LS を適用した実験結果

DCOMP-LS を C 言語を用いてプログラム化し，ISCAS'89 ベンチマーク回路に対して行った実験の結果を示す．まず，元のテスト系列として HITEC によって生成されたテスト系列を用いた実験を行い，DCOMP-S と DCOMP-LS の結果を比較する．ここでは，DCOMP-LS 中の変数 L を，元のテスト系列長の 50%とした．**表 8** に実験の結果を示す．表中の "S"，"LS" の列にはそれぞれ，DCOMP-S，DCOMP-LS の結果を示す．DCOMP-S の結果は表 7 に示されるものと同一である．"compacted vectors" の列には，圧縮されたテスト系列長を示し，"%" の列には，元のテスト系列長に対する，元のテスト系列から削減されたテストベクトル数の割合を示す．"cpu(s)" には計算時間を示す．これらの結果より，すべての回路に対して，DCOMP-LS によってテスト系列を圧縮することができたが，DCOMP-S に比べて削減されたテストベクトル数は少ないことが分かる．DCOMP-S が約 10%から 72%のテストベクトルを削減できたのに対し，DCOMP-LS は約 3%から 35%のテストベクトルしか削減することができなかった．しかしながら計算時間については，多くの回路において DCOMP-LS の方が短かった．ただし s382，s400，s444，s526 回路では，DCOMP-S の計算時間が DCOMP-LS より短かった．これは，DCOMP-S の方が多くのテストベクトルを削減できたため，テスト系列長が短くなり，その結果，故障シミュレーション時間が短縮したためと考えられる．

表 9 に s35932 回路に対して DCOMP-LS を適用した結果を示す．変数 L は，元のテスト系列長の 80%および 90%とした．テスト系列として，3 種類の系列を用いた．

- HITEC によって生成されたテスト系列（表中では H と表示）
- HITEC によって生成されたテスト系列に 50 個のランダムベクトルを追加した系列（表中では H+R と表示）
- 初期化ベクトルとランダムベクトルのみで構成された長さ 1,024 の系列（表中では R と表示）

表中の type，L(%) の各列には，テスト系列の種類，変数 L の割合を示す．また，cov(%)，len，comp，%，pair，undis，cpu(s) の各列は，表 6 と同様のデータ

を示す．実験の結果，HITEC のみのテスト系列ではテストベクトルを削減することができなかったが，これは HITEC によるテスト系列には診断に関して冗長なテストベクトルが，元のテスト系列の最後の 10%および 20%の部分にはあまり含まれていなかったためと考えられる．一方，ランダムベクトルを追加したテスト系列では 2.0%，初期化ベクトルとランダムベクトルのみのテスト系列では 15.7%のテストベクトルを削減することができ，提案法が有効であることが分かる．また，DCOMP-LS では s35932 のように 6 億以上の故障ペアを持つ回路に対してもテスト圧縮が可能であることが示された．この実験では変数 L を 80%および 90%とした．一般的には，故障診断に関して冗長なテストベクトルがテスト系列中のどの部分に多く含まれているかは未知であるため，L の値を小さくした場合に，計算時間は増大するが，削除可能なテストベクトルの探索範囲が広くなり，より多くのテストベクトルが削減されると考えられる．しかしながら，HITEC テスト系列にランダムベクトルを追加した系列のように，L が大きい場合により多くのテストベクトルが削除されることがある．これは，$L = 80\%$ の場合と $L = 90\%$ の場合では状態遷移が異なっており，元と同じ故障ペアを区別するために $L = 80\%$ の場合に，より多くのテストベクトルが必要となったためと考えられる．L の値は計算時間と結果として得られるテスト系列長に大きく影響するため，最適な L を求めることが今後の課題として残されている．

5. おわりに

本論文では組合せ回路および順序回路に対する診断用テスト集合およびテスト系列を圧縮する手法を提案した．組合せ回路に対するテスト圧縮では，一部の故障ペアに対して故障区別表を作成し，それを元にテストベクトルを選択する手法を繰り返し適用した．これによって大規模回路においてもテスト圧縮を可能にした．

順序回路に対するテスト圧縮では，逆順ベクトル回復法をもとにテスト圧縮を行った．大規模回路では故障ペア数が膨大であるため，これを緩和する発見的手法を導入することで大規模回路においてもテスト圧縮を可能にした．しかしながら，他の回路に比べて大規模回路においては十分短いテスト系列が得られたとはいえず，今後さらに短いテスト系列を得るような手法を開発する予定である．

謝辞 本研究は一部，日本学術振興会科学研究費補助金（基盤研究（C）15500043）の援助による．

参考文献

1) 樋上喜信，梶原誠司，市原英行，高松雄三：論理回路に対するテストコスト削減法―テストデータ量およびテスト実行時間の削減，信学会論文誌 D-I，Vol.J87-D-I, No.3, pp.291–307 (2004).
2) 高橋 寛, 山本幸大, 樋上喜信, 高松雄三：BIST 環境における不確かなテスト集合による単一縮退故障の一診断法，信学会論文誌 D-I，Vol.J88-D-I, No.6, pp.1029–1038 (2005).
3) Boppana, V. and Fuchs, W.K.: Dynamic fault collapsing and diagnostic test pattern generation for sequential circuits, *Dig. Int. Conf. on Computer-Aided Design*, pp.147–154 (1998).
4) Guo, R., Pomeranz, I. and Reddy, S.M.: On speed-up vector restoration based static compaction of test sequences for sequential circuits, *Proc. Asian Test Sympo.*, pp.467–471 (Dec. 1998).
5) Hartanto, I., Boppana, V., Patel, J.H. and Fuchs, W.K.: Diagnostic test pattern generation for sequential circuits, *Proc. VLSI Test Symp.*, pp.196–202 (1997).
6) Niermann, T.M. and Patel, J.H.: HITEC: A test generation package for sequential circuits, *Proc. European Conf. on Design Automation*, pp.214–218 (Feb. 1991).
7) Pomeranz, I., Venkataraman, S. and Reddy, S.M.: Z-DFD: Design-for-diagnosability based on the concept of Z-detection, *Proc. Int. Test Conf.*, pp.489–497 (2004).
8) Rudnick, E., Fuchs, W. and Patel, J.: Diagnostic fault simulation of sequential circuits, *Proc. Int. Test Conf.*, pp.178–186 (1992).
9) Shao, Y., Guo, R., Pomeranz, I. and Reddy, S.M.: The effects of test compaction on fault diagnosis, *Proc. Int. Test Conf.*, pp.1083–1089 (1999).
10) Venkataraman, S., Fuchs, W.K. and Patel, J.H.: Diagnostic Simulation of Sequential Circuits Using Fault Sampling, *Proc. Int. Conf. on VLSI Design*, pp.476–481 (1998).

(平成 17 年 10 月 20 日受付)
(平成 18 年 4 月 4 日採録)

樋上　喜信（正会員）

1996 年大阪大学大学院工学研究科応用物理学専攻博士後期課程修了．同年日本学術振興会特別研究員採用．1998 年より愛媛大学工学部助手．現在，同学部助教授．論理回路に対するテスト生成およびテスト容易化設計に関する研究に従事．電子情報通信学会，IEEE 各会員．博士（工学）．

ケーワル K. サルージャ

1967 年ルーキー大学電気卒業．1972 年アイオワ大学大学院修士課程修了．1973 年アイオワ大学大学院博士課程修了．オーストラリア，ニューキャッスル大学を経て，現在ウィスコンシン大学マディソン校電気コンピュータ工学科教授．論理設計，コンピュータアーキテクチャ，マイクロプロセッサシステム，VLS1 設計とテストに関する教育に従事．これまで，テスト生成，故障シミュレーション，テスト容易化設計，組み込み自己テストのツールやアルゴリズムを開発．現在，順序回路テスト生成，テスト容易化合成，組み込み自己テスト，順序回路検証の研究に従事．国際会議，論文誌等において 200 以上の論文を発表．南カリフォルニア大学，アイオワ大学，広島大学，奈良先端科学技術大学院大学，ルーキー大学客員教授．合衆国開発プログラム顧問．IEEE Trans. on Computer 編集委員．JETTA レター部門編集委員．JSPS フェロー．IEEE フェロー．

高橋　　寛（正会員）

1988 年佐賀大学理工学部電子工学科卒業．1990 年同大学大学院理工学研究科修士課程修了．愛媛大学工学部助手，講師を経て，2000 年より同大学助教授，現在に至る．2000 年 5 月〜2001 年 3 月までウィスコンシン大学客員研究員（文部科学省在外研究員）．論理回路のテスト生成および故障診断に関する研究に従事．博士（工学）．電子情報通信学会，IEEE 各会員．

小林　真也（正会員）

1985 年大阪大学工学部情報工学科卒業．1991 年同大学大学院博士課程修了．工学博士．同年金沢大学工学部電気・情報工学科助手．その後，同講師，助教授，同大学大学院自然科学研究科助教授を経て，1999 年愛媛大学工学部情報工学科助教授．2004 年同学科教授．2006 年より，愛媛大学大学院理工学研究科電子情報工学専攻教授．その間 1996 年カリフォルニア大学アーバイン校客員教員．2002〜2003 年ワシントン大学バセル校客員研究員．並列処理システム，分散処理システムの研究に従事．電子情報通信学会，日本ソフトウェア科学会，電気学会，IEEE，ACM 各会員．訳書『計算機設計技法』（トッパン），著書『基礎から学ぶ UNIX ワークステーション』（トッパン）．

高松　雄三（正会員）

1966 年愛媛大学工学部電気工学科卒業．佐賀大学理工学部電子工学科助教授を経て，1987 年 10 月より愛媛大学工学部情報工学科教授，現在に至る．論理回路の診断・テストに関する研究に従事．工学博士．著書『論理設計入門』（共著，日新出版）等．電子情報通信学会正員，IEEE Senior Member．