Deterministic BISTのためのテストデータ圧縮の一手法

林 照峯[†] 鈴木孝宣[†]
戸田隆宏[††] 篠木剛[†]

本論文では，ROMに貯えられたテストデータを1ビットずつシフトして被検査組合せ回路（CUT）のテストパターンとするdeterministic BIST方式において，コンパクトなテストデータを作成する有効な一手法を提案する．提案手法は，C1-compatibilityを用いた回路入力幅圧縮後のCUTに対してコンパクトなATPGベクトル集合の生成を行った後，この集合をもとにしてコンパクトなテストデータを作成する．そのために提案手法では，(1) ATPGベクトルだけでなくシフトベクトルでの故障検出も考慮した故障検出困難度を用いてROMに格納するATPGベクトル数を小さくするとともに，(2) 不要ビット削除に基づくビット単位でのテストデータ圧縮を行っている．提案手法の有効性をISCASベンチマーク回路に対する評価実験によって示す．とくに，c2670やc7552等，乱数パターンで検出しにくい故障を多く持つ回路に対して大きな効果があることを示す．

A Test Data Compaction Method for Deterministic BIST

Terumine Hayashi,[†] Takanori Suzuki,[†] Takahiro Toda[††]
and Tsuyoshi Shinogi[†]

This paper presents a compact test data generation method for combinational circuits-under-test (CUTs) with a deterministic BIST structure, under which test vectors to the CUT are produced by shifting ROM test data one by one bit. The proposed method generates compact ROM test data from a compact ATPG vector set generated for the CUT after input width compression with C1-compatibility. The features are as follows. (1) ATPG vectors to be included in ROM test data are reduced by using hard-to-detect measures which take account of fault detection by not only ATPG vectors but also shift vectors, and (2) unnecessary bits in ROM test data are removed for further compression. It is shown that the proposed method is effective to generate compact test data with 100% fault efficiency through experiments for ISCAS benchmark circuits. Especially, it is very effective for circuits with many random-pattern-resistant (*rpr*) faults such as c2670, c7552 and so on.

1. はじめに

将来的にはテスタスピードの向上がLSIデバイスの性能向上に追いつかなくなるとの予想がある．また，LSI設計技術の進歩よりも製造技術の進歩の方が早く，LSI内部にテスト機構を内蔵する十分な余裕ができてくるとの見方がある．このような背景から，ある程度の面積オーバヘッドの存在にもかかわらず，組込み自己テスト（BIST: Built-in Self Test）は今まで以上に重要となり，その実用化も進展していくものと思われる．

BISTとしては，LFSR（Linear Feedback Shift Register）を用いて擬似乱数パターンを発生させる方式がよく用いられている．この方式は，構成のしかたによってはat-speed testが可能になり，また比較的簡単な構造で多くのパターンを発生できるという利点がある．しかし，被検査回路（CUT: circuit under test）内の乱数パターンでは検出されにくい *rpr*（random pattern resistant）故障の多くが未検出のまま残ってしまうという問題がある．また，テストサイクル数☆が大きくなるため，テスト時の消費電力増大等が問題となることもある．

LFSRベースのBISTにおいて，故障検出率を向上させる手法がすでにいくつか提案されている．文献1)

† 三重大学工学部電気電子工学科
Department of Electrical and Electronic Engineering, Faculty of Engineering, Mie University
†† 株式会社富士通コンピュータテクノロジー
Fujitsu Computer Technology Limited

☆ 本論文では，CUTに印加されるテスト入力ベクトルの数をテストサイクル数と呼ぶ．

では，組合せ回路用のテスト生成システム（ATPG: Automatic Test Pattern Generator）により生成されたテストベクトル（単に ATPG ベクトルと略す）を LFSR の初期値として用いる reseeding 法と，LFSR の構成を変えるための多項式選択法を組み合わせることによって *rpr* 故障を検出できるようにし，故障検出率を向上させている．文献 2) では LFSR により発生された乱数パターンでの未検出故障を検出するようにテストポイントを挿入する方法が提案されている．

LFSR を用いない方法として，文献 3) では ROM に格納されたテストデータを k 分割して CUT の入力レジスタにシフト入力する構成のテストパターン生成器において，ATPG ベクトル集合を利用して ROM テストデータを作ることによって単一縮退故障に対する故障検出効率☆100%を得る方法が示されている．また，文献 4) では，部分ローテート型スキャン回路を用いた構造により，テストデータ量を減らしている．しかし，これらの方法を用いても，*rpr* 故障を多く含む CUT 等ではテストデータ量（すなわち，面積オーバヘッド）が大きくなってしまうことがある．

一方，回路入力間の compatibility の概念を用いたカウンタベースの BIST 方式[5] も提案されている．この方式は，回路入力の 2 つをそのまま（あるいは否定して）結線したとしても，回路内の検出可能な故障をすべて検出できるときには，それら 2 つの回路入力を 1 つの回路入力にまとめることにより，テスト時の回路入力数を減らしたうえで網羅的に全パターンを CUT に入力することを基本としている．また，文献 6) では，文献 5) で提案された compatibility の概念を C1-compatibility と呼んでこれを Ck-compatibility に拡張するとともに，C2-compatibility に基づいて回路入力数をさらに減らす手法を提案している．さらに，文献 7) では，SCk-compatibility への拡張と，故障集合のグループ分けに基づく方法も提案されている．これらカウンタベースの BIST 方式は故障検出効率 100%を網羅的パターンで達成できるようにしており，また面積オーバヘッドも比較的小さく有効であるが，LFSR ベースの BIST と同様にテストサイクル数が大きくなりがちである．

ここでは，文献 1)，3)，4) の BIST のように何らかの形で ATPG ベクトルでの故障検出を利用する BIST を deterministic BIST と呼ぶことにする．deterministic BIST の特徴は，故障検出率を高くすることが容易であり，また，テストサイクル数も他の BIST 方式に比べて小さくできる点にある．しかし，ATPG ベクトルを ROM テストデータとして貯えるときには，そのデータ量をいかに小さく抑えるかが重要となる．

本論文では，ROM に格納されたテストデータを 1 ビットずつシフトする deterministic BIST 構造[3] において，テストデータ量を小さくするという意味において最も有利な $k = 1$（すなわち分割なし）の場合を対象として，単一縮退故障検出効率 100%の条件を満足するさらにコンパクトなテストデータの作成手法を提案する．

提案する手法は，C1-compatibility 処理により圧縮された回路入力を持つ CUT に対するコンパクトな ATPG ベクトル集合の生成と，故障検出困難度に基づく評価値を用いて ATPG ベクトルを選んで並べることを基本とするテストデータ生成，および，ビット単位でのテストデータ圧縮，を組み合わせることによって *rpr* 故障を多く含む回路に対してもコンパクトな ROM テストデータを実用的な処理時間で作成できるようにすることを主眼としている．

本論文では，2 章で BIST の基本構成を述べ，3 章で提案手法を述べた後，4 章で ISCAS ベンチマーク回路[10),11)] に対する実験結果を述べ，提案手法の有効性を示す．5 章では本論文のまとめを述べる．

2. BIST の基本構成と C1-Compatibility

2.1 BIST の基本構成

対象とする BIST の基本構成を**図 1** に示す．ROM（RAM でも可）に格納されたテストデータは 1 ビットずつ入力シフトレジスタ（SR）にシフト入力され，検査対象回路（CUT）のテストベクトルとなる．ここでは CUT は組合せ回路であるものとする．一方，CUT からの応答出力はシグネチャ解析器等の応答解析器（RA）に圧縮して記憶される．故障の有無の判

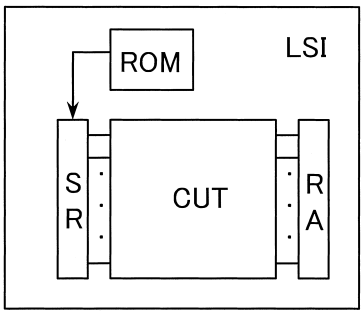


図 1 BIST の基本構成
Fig. 1 Basic BIST structure.

☆ 対象となる全故障数に対する，検出された故障数と検出不能な故障数の和の割合を故障検出効率と呼ぶ．

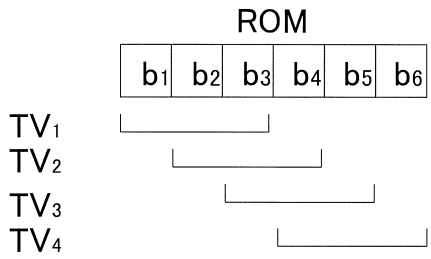


図 2 CUT へのテストベクトル
Fig. 2 Test vectors to CUT.

定は，RA に最終的に格納された値を外部から観測することによって行われる．

図 2 に ROM に格納されたテストデータと，CUT へのテストベクトルの関係を例で示す．この例では，b_1 〜b_6 の 6 ビットのテストデータが ROM に格納されており，CUT が 3 入力であるとき，CUT へは，$TV_1 = (b_3, b_2, b_1)$，$TV_2 = (b_4, b_3, b_2)$，$TV_3 = (b_5, b_4, b_3)$，$TV_4 = (b_6, b_5, b_4)$ の 4 つのテストベクトルが印加されることを表している．

ROM に格納するテストデータとしては，CUT の組合せ回路に対して生成された ATPG ベクトル集合に含まれる ATPG ベクトルを 1 列に並べて作る方法がある．たとえば，上記の例で ATPG ベクトルとして ATPG_V1 $= (b_3, b_2, b_1)$ と ATPG_V2 $= (b_6, b_5, b_4)$ の 2 つを選び 1 列に並べると，図 2 の ROM テストデータとなる．この場合，TV_1 と TV_4 はそれぞれ ATPG ベクトル ATPG_V1 と ATPG_V2 そのものであり，そのほかにテストベクトル $TV2$ と $TV3$ が出現する．このようなテストベクトル TV_2，TV_3 をシフトベクトルと呼ぶ．

この方式では，ATPG ベクトルを用いるので，LFSR 等の擬似乱数発生による方法よりも故障検出効率 100% の達成が容易である．また，シフトベクトルによる故障検出を考慮するので，すべての ATPG ベクトルを ROM に格納する必要がなく，比較的コンパクトな ROM テストデータを作りやすいという利点がある．また，LFSR ベースの BIST や，カウンタベースの BIST に比べて，CUT に印加されるテストベクトル数（テストサイクル数）も少なくて済む．しかし，ROM に格納するテストデータをいかに小さくするかがこの方式のポイントでありテストデータ量により，面積オーバヘッド，テストサイクル数とも直接影響を受ける．

本論文では，この BIST 構造を持つ CUT に対するコンパクトな ROM テストデータの生成手法を提案することを目的とする．

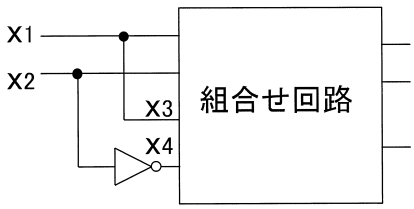


図 3 C1-compatibility の例
Fig. 3 Example of C1-compatibility.

2.2 C1-compatibility

回路入力間の compatibility の概念はカウンタベース BIST における網羅パターン生成を可能にするために文献 5) で導入され，文献 6) で Ck-compatibility に拡張された．組合せ回路のある回路入力 x を外部から入力する代わりに，他のいくつかの回路入力 $x_1, x_2, \cdots, x_k$ を変数とする論理関数 $x = f(x_1, x_2, \cdots, x_k)$ によって与えても，もとの組合せ回路内の検出可能な故障をすべて検出できるとき，回路入力 x と $(x_1, x_2, \cdots, x_k)$ は Ck-compatible であるという．

ここでは，C1-compatibility のみを用いる．すなわち，ある回路入力 x_j が x_i の論理関数，すなわち $x_j = x_i$ または $x_j = \bar{x}_i$ とすることができる場合のみを考える．たとえば，**図 3** の例では，$x_3 = x_1$，$x_4 = \bar{x}_2$ とすることにより，もとの回路入力が 4 個であったのを，実質的に 2 個の独立な回路入力にできるということを示している．

このように，C1-compatibility の概念を利用すると実質的にテスト時の回路入力数を減らすことができるので，故障検出効率 100%の達成に要する ATPG ベクトル数が C1-compatibility を用いないときに比べてあまり増加しない場合には，結果としてテストデータ量を減らすことができる．

本論文では，前節で述べた deterministic BIST におけるコンパクトな ROM テストデータの生成のために C1-compatibility の概念を利用する．

3. 提 案 手 法

3.1 提案手法の概要

提案手法は C1-compatibility 処理に基づく回路入力方向のテストデータ圧縮と，シフトベクトルでの故障検出を利用したテストベクトル方向の圧縮からなっており，以下の 4 段階の手順からなる．

〔コンパクトな ROM テストデータ生成手法〕

［処理手順 1］（C1-compatibility 計算）

CUT の中の C1-compatible な回路入力ペア x_i と x_j を求める．この処理は回路構造に基づく方法と，テス

ト生成に基づく方法からなる．回路構造に基づく方法では，x_i と x_j が共通のファンアウト領域を持つかどうかを調べ，もし，共通のファンアウト領域を持たないならば，x_i と x_j はお互いに依存関係になく仮に 1 つの信号線として扱ったとしても故障検出可能性に影響を与えないので，x_j を x_i に従属する単なる信号線と見なす．テスト生成に基づく方法では，決定論的テスト生成手法の処理技法を利用して C1-compatible な回路入力ペア x_i と x_j を求め，x_j を x_i に従属する単なる信号線と見なす．この処理を C1-compatible な回路入力ペアがなくなるまで行う．これには文献 6) の手法を用いることができる．

［処理手順 2］（組合せ回路のコンパクト ATPG ベクトル集合の生成）

C1-compatibility 処理後の CUT に対して，組合せ回路用のコンパクト ATPG ベクトル集合の生成を行い，ATPG ベクトル集合を作成する．これには文献 8)，9) に代表される組合せ回路用のコンパクト ATPG ベクトル集合生成アルゴリズムを用いることができる．

［処理手順 3］（ATPG ベクトル単位での ROM テストデータ生成）

ATPG ベクトル集合から，シフトベクトルでの故障検出も考慮に入れて ATPG ベクトルを選び 1 列に並べ，故障検出効率 100%の ROM テストデータを作る．この処理については，3.2 節で具体的に説明する．

［処理手順 4］（ビット単位での ROM テストデータ圧縮）

ROM テストデータの各ビットについて，そのビットを取り除いても故障検出効率 100%を達成できるかどうか確かめる．もし，達成できるならそのビットを削除する．この処理を，どのビットも削除できなくなるまで繰り返す．この処理については，3.3 節で具体的に説明する．

3.2 ATPG ベクトル単位でのテストデータ生成

ROM テストデータは ATPG ベクトルを 1 列に並べることによって構成され，CUT には ATPG ベクトルとシフトベクトルが入力される．シフトベクトルはほぼランダムパターンであるため，*rpr* 故障の多くは ATPG ベクトルでの検出に頼ることになる．しかし，シフトベクトルでの故障検出も重要であり，ATPG ベクトルとシフトベクトルを合わせて検出困難な故障を多く検出できるように ATPG ベクトルを選ぶ必要がある．これを反映するため，提案手法ではまず，各故障 f の検出困難度 $w(f)$ を下記で定義する．

$$w(f) = 1/N_{det}(f) \tag{1}$$

ここで，$N_{det}(f)$ はある入力パターン集合を CUT に入力したときの故障検出回数である．その入力パターン集合としては，処理手順 2 で求められたコンパクト ATPG ベクトル集合に一定数の乱数パターンを追加したものを用いる．CUT 内のすべての故障に対して故障検出困難度をあらかじめ計算しておく．

次に，この故障検出困難度を用いて，各 ATPG ベクトル V の故障検出能力を表す評価値 $E(V)$ を次のように定義する．

$$E(V) = \sum_{f \in F_v \subseteq F\_SET} w(f) \tag{2}$$

ここで，F_SET は未検出故障集合であり，F_v は F_SET に含まれる故障の中で ATPG ベクトル V によって検出される故障の集合を表している．定義から，$E(V)$ の値の大きな ATPG ベクトルほど検出困難な故障を多く検出するのに有効であり，ROM テストデータに含めた方がよいと見なすことができる．

さらに，2 つの ATPG ベクトル V_i の後に V_j を隣接して並べて ROM データを作るときに現れるシフトベクトルの評価値 $E_{shift}(V_i, V_j)$ を次式で定義する．

$$E_{shift}(V_i, V_j) = \sum_{f \in F(V_i, V_j) \subseteq F\_SET} w(f) \tag{3}$$

ここで，$F(V_i, V_j)$ は，V_i と V_j によって作られるシフトベクトルと，V_i および V_j によって新たに検出される故障集合を表す．

本論文では，これらの評価値を用いて，検出困難な故障をできるだけ多く検出するようなテストデータを greedy に求める手法を示す．すなわち，本手法では，まず評価値 $E(V)$ が最大の ATPG ベクトルをテストデータに含めたのち，故障検出効率が 100%になるまで，**図 4** の (1),(2) で示すようにその時点までに作られたテストデータ $V^{(1)}V^{(2)}\cdots V^{(N)}$ の後または前に，評価値 $E_{shift}(V^{(N)}, V')$ または $E_{shift}(V', V^{(1)})$ の大きい ATPG ベクトルを追加していくことを繰り

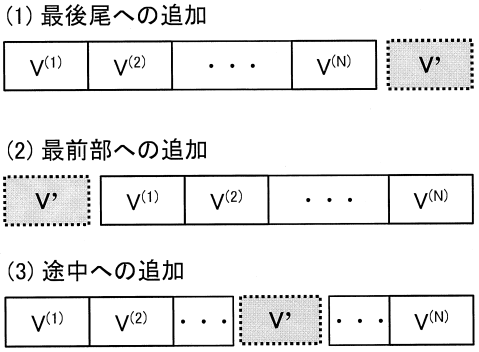


図 4 ATPG ベクトルの追加
Fig. 4 Attachment of ATPG vector.

返す．ここで，新たな ATPG ベクトルを，テストデータの途中に挿入（図 4 の（3））することをせずに，後と前への追加のみに限定したのは，追加されるべき ATPG ベクトルと，追加によって現れるシフトベクトルを故障シミュレーションするだけで，新たに検出される故障を求めることができるからである．一方，テストデータの途中に ATPG ベクトルを挿入すると，挿入された部分の前後のシフトベクトルも変わってしまうため，結局，挿入後のテストデータによって作られる CUT のテストベクトルすべてを再度故障シミュレーションしなければならず，処理時間が大となる．

以下に，ATPG ベクトル単位でのテストデータ生成手続きを示す．

[ATPG ベクトル単位でのテストデータ生成手続き]

入力：故障検出効率 100%の ATPG ベクトル集合 *TP_SET*，CUT 内の検出可能な全故障の集合 *ALLF_SET*．

出力：N 個の ATPG ベクトル $V^{(1)}, V^{(2)}, \cdots, V^{(N)}$ を順に並べた ATPG ベクトル列 $V^{(1)}V^{(2)}\cdots V^{(N)}$ によって作られる故障検出効率 100%の ROM テストデータ．

（Step 1） CUT 内のすべての検出可能な故障に対して故障検出困難度 $w(f)$ を計算する．現時点での未検出故障集合を *F_SET* で表し，$F\_SET \leftarrow ALLF\_SET$ とする．

（Step 2） $E(V)$ の値が最も大きい ATPG ベクトル V を求める．$N \leftarrow 1$，$V^{(1)} \leftarrow V$，$TP\_SET \leftarrow TP\_SET - \{V\}$ とし，ベクトル V によって検出される故障を *F_SET* から除去する．

（Step 3） *TP_SET* に含まれる各 ATPG ベクトル V について，評価値 $E_{shift}(V^{(N)}, V)$ と $E_{shift}(V, V^{(1)})$ を計算し，最も大きな評価値を持つ ATPG ベクトルを V' とする．最大の評価値が $E_{shift}(V^{(N)}, V')$ で与えられるときには Step 4 へ，$E_{shift}(V', V^{(1)})$ で与えられるときには Step 5 へ．

（Step 4） $N \leftarrow N+1$ とし，$V^{(N)} \leftarrow V'$，$TP\_SET \leftarrow TP\_SET - \{V'\}$ とする（図 4 の（1）参照）．$V^{(N-1)}$ と $V^{(N)}$ を並べたときに作られるシフトベクトルおよび $V^{(N)}$ によって新たに検出される故障を *F_SET* から除去する．$F\_SET = \phi$ なら手続き終了，さもなければ Step 3 へ．

（Step 5） ATPG ベクトル列 $V^{(1)}V^{(2)}\cdots V^{(N)}$ を $V^{(2)}V^{(3)}\cdots V^{(N+1)}$ とする．$N \leftarrow N+1$ とし，$V^{(1)} \leftarrow V'$，$TP\_SET \leftarrow TP\_SET - \{V'\}$ とする（図 4 の（2）参照）．$V^{(1)}$ と $V^{(2)}$ を並べたときに作られるシフトベクトルおよび $V^{(1)}$ によって新たに検出される故障を *F_SET* から除去する．$F\_SET = \phi$ なら手続き終了，さもなければ Step 3 へ．

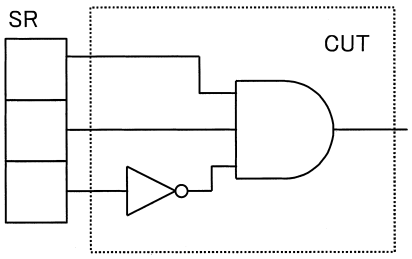


図 5 3 入力 CUT の例
Fig. 5 Example of CUT with 3 inputs.

以上述べた手続きによって，与えられた ATPG ベクトル集合 *TP_SET* から，N 個の ATPG ベクトルを選んで 1 列に並べたテストデータが作成される．このときのテストデータ量は $N \cdot M$ ビットである．ここで，M は C1-compatibility 処理後の回路入力数である．

3.3 ビット単位でのテストデータ圧縮

3.2 節で述べた ATPG ベクトル単位でのテストデータ生成手続きによって故障検出効率 100%の ROM テストデータを作成することができる．いま，ROM テストデータが r ビットのビット列 $b_1b_2b_3\cdots b_r$ として得られているものとする．このビット列の中には，削除しても故障検出効率が低下しないビットをいくつか含んでいることが多い．たとえばいま，**図 5** に示す 3 入力の CUT において，故障検出効率 100%を達成するために 4 つの ATPG ベクトル ATPG_V1 = (1,0,0)，ATPG_V2 = (0,1,0)，ATPG_V3 = (1,1,1)，ATPG_V4 = (1,1,0) が得られており，このうち，ATPG_V1，ATPG_V2，ATPG_V3 の 3 つを用いて 9 ビットの ROM テストデータ 001010111 が作られているものとする．このテストデータのたとえば 3 ビット目と 4 ビット目を削除して，7 ビットのテストデータ 0010111 としても故障検出効率 100%を達成できる．このことは，不要なビットを削除することによって，さらにテストデータをコンパクトにできることを示している．

あるビットが不要ビットであるかどうかは，仮にそのビットを削除したテストデータを作ってみて，故障シミュレーションを行い故障検出効率が低下しないかどうか調べるという単純な方法によって判定することができる．しかし，この方法では 1 つのビットの削除可否を調べるのに故障シミュレーションを r 回（r パターン分）行うことになので，全体として故障シミュレーションをほぼ r^2 に比例する回数行うことが必要

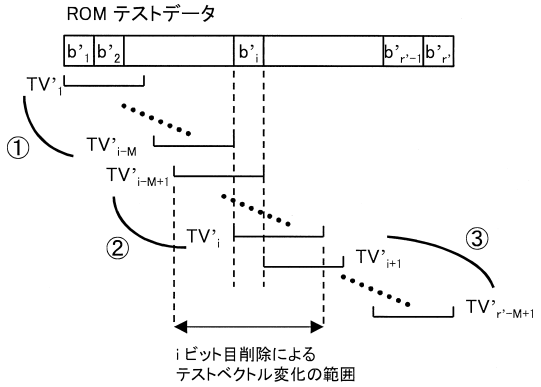


図 6 ビット削除によるテストベクトルの変化
Fig. 6 Change of test vectors by bit removal.

となる．このことはテストデータ長の大きな大規模回路では処理時間が膨大になり，実用的でなくなることを意味している．

この問題を解決するため，我々はある 1 つのビットの削除可否を調べるための故障シミュレーションを r に依存しない回数で済ませることにより，全体の故障シミュレーションを r に比例する回数で済ませることのできる方法を提案する．

いま，テストデータの 1 ビット目から順に削除可否をチェックし，削除可能ならそのビットを削除し，そうでなければそのビットを残すという処理を行っていくものとする．そして，$(i-1)$ ビット目までの削除可否チェック，および，削除可能ならばそのビットの削除が行われた後のテストデータ $b'_1b'_2b'_3\cdots b'_{r'}$（ただし，$r' \leq r$）に対して，$i$ ビット目の削除可否チェックを行おうとしているものとする．また，このときのテストデータによる CUT のテストベクトルを**図 6** に示すように $TV'_1, TV'_2, \cdots, TV'_{r'-M+1}$ で表すことにする．これらのテストベクトルを図 6 のように，① $TV'_1, \cdots, TV'_{i-M}$ と ② $TV'_{i-M+1}, \cdots, TV'_i$ と ③ $TV'_{i+1}, \cdots, TV'_{r'-M+1}$ の 3 つの部分に分けると，① と ③ は，i 番目のビットを仮に削除したときの CUT へのテストベクトルでもあるので，結局 ② の部分のみが i ビット目の削除前と後とで異なることになる．いま，② の部分が i ビット目の削除によって $TV''_{i-M+1}, \cdots TV''_{i-1}$ に変わるものとし，この部分を ②' とする．

$(i-1)$ ビット目の削除可否チェック処理が終了しているということに着目して，①②'③ の各部分で検出できる故障を下記により計算する．

（a）① の部分の故障検出

$(i-1)$ ビット目の削除可否チェック処理が終了した時点で，$TV'_1, \cdots, TV'_{i-M-1}$ によって検出される故障の情報は得られているので，その情報を利用すれば故障シミュレーションを行わずに済ませることができる．一方，TV'_{i-M} は $(i-1)$ ビット目の削除可否チェック処理では現れないテストベクトルであるので故障シミュレーションを行い，検出する故障を求める．結局，この部分の故障シミュレーション回数は 1 回のみである．

（b）②' の部分の故障検出

この部分は i ビット目の削除によってはじめて現れるテストベクトルであるので，そのすべてについて故障シミュレーションを行う必要がある．この部分の故障シミュレーション回数は $(M-1)$ 回である．M の値は CUT の構造によって変わるが，r とは直接関係しないので，定数と見なすことができる．結局，故障シミュレーションを r に依存しない回数に減らすことができる．

（c）③ の部分の故障検出

① と ②' のテストベクトルで検出されなかった故障が，③ のテストベクトルで検出されるかどうかを計算する必要がある．③ のテストベクトル自体は 1〜$(i-1)$ ビット目の削除処理によって変わることはないが，① と ②' の部分が変わってしまうので，③ の部分で検出できるかどうかの計算対象の故障が変わってしまう．したがって，故障ドロップ（検出された故障をシミュレーション対象から外すこと）を用いる通常の故障シミュレーションを用いる場合には再度シミュレーションやり直す必要があるように見える．しかし，我々は以下の方法によって，③ の部分で検出される故障を故障シミュレーションの再実行なしに求めることができるようにする．まず，ビット削除可否処理全体の前処理として，あらかじめ初期テストデータ $b_1b_2b_3\cdots b_r$ から得られるテストベクトル $TV_1, TV_2, \cdots, TV_{r-M+1}$ を逆順に，すなわち $TV_{r-M+1}, \cdots, TV_2, TV_1$ の順に故障シミュレーションを行い，すべての故障 f について何パターン目ではじめて検出されたのかを表す値 $det(f)$ を求めておく．すなわち，故障 f が j パターン目のテストベクトル $TV_{r-M+2-j}$ ではじめて検出されたときには $det(f) = j$ とおく．

このようにして前処理で求めておいた情報を利用すると，$det(f) \leq (r'-i-M+1)$ なら故障 f は ③ の部分によって検出され，そうでなければ検出されないと判定できる．なぜなら，③ の部分のパターン数は $(r'-i-M+1)$ であり，しかもこの部分はもとのテストデータと同じままで変わらないからである．したがって，③ の部分による故障検出の計算を故障シ

ミュレーションなしで行うことができる．

図 7 に，ビット単位でのテストデータ圧縮手続きを示す．

4. 実験結果

3 章で述べた手法の効果を確認するため，ISCAS ベンチマーク回路[10),11)] に対して評価実験を行った．ただし，ISCAS'89 回路については，その組合せ回路部分を CUT と見なして処理を行った．まず**表 1** は予備実験であり，C1-compatibility 処理を行った場合と行わない場合について，組合せ回路用のコンパクト ATPG ベクトル集合生成システムを利用して回路入力数と故障検出効率 100%を得るための ATPG ベクトル数の関係を求め，全 ATPG ベクトルを ROM に格納したときの容量を調べたものである．なお，ISCAS'85 回路については，これと同等の実験結果が文献 5) の中で示されている．また，C1-compatibility 処理については，文献 6) の中で実用的な時間で処理できるアルゴリズムが示されている．本予備実験により，C1-compatibility 処理を行うことによって，回路入力数は約 1/3〜1/250 に減ったのに対し，ATPG ベクトル数はたかだか 2 倍程度以下，平均で 1.4 倍程度しか増加せず，結局，回路入力数 × ATPG ベクトル数では，平均で約 1/5 程度に減らすことができるという結果が得られた．このことは，C1-compatibility 処理による ATPG ベクトル数増加のペナルティよりも，回路入力数減少の効果の方がはるかに大きいということを示している．

表 2 は，提案手法によるコンパクトな ROM テストデータの生成実験結果を示している．ここで，表中の $Na \cdot Ma$，$Nb \cdot Mb$ は表 1 のものと同じであり，それぞれ，C1-compatibility 処理を行わない場合と行った場合のコンパクト ATPG ベクトル集合に含まれる全 ATPG ベクトルをそのまま ROM テストデータとして格納したときの総ビット数を表している．実験により，提案手法はテストデータ量を $Na \cdot Ma$ に比べて 17%以下，平均で約 9%に，また，$Nb \cdot Mb$ に比べて 30〜60%程度に圧縮できた．さらに，ROM ビット数のゲート換算を文献 3) に従い，5.65 ビット ＝ 1 ゲートとしたときのテストデータの面積オーバヘッドを CUT の規模に対する割合として算出した数値を表の O.H. の欄に載せた．もとの回路のゲート数として，本来は実装ゲート数を用いるべきであり，また CUT 内の論理ゲートだけでなく，入力レジスタ，出力レジスタ，デコーダ等も考慮に入れるべきではあるが，こ

〔ビット単位のテストデータ圧縮手続き〕
入力：圧縮前 r ビットテストデータ $b_1,b_2,...,b_r$
出力：圧縮後 $r'(r' \leqq r)$ ビットテストデータ $b'_1,b'_2,...,b'_{r'}$
begin
　逆順故障シミュレーションにより各故障 f の det(f) を計算；
　$i \leftarrow 1$，$r' \leftarrow r$；
　$b'_1,b'_2,...,b'_{r'} \leftarrow b_1,b_2,...,b_r$；
　while($i \leqq r'$) {
　　i ビット目を仮に削除し，検出できる故障を計算；
　　if 故障検出効率100%
　　　then i ビット目を削除し，$r' \leftarrow r'-1$；
　　　else $i \leftarrow i+1$；
　}
end

図 7 ビット単位のテストデータ圧縮手続き
Fig. 7 Bit-wise test data compression procedure.

表 1 C1-compatibility と ATPG ベクトル数の関係
Table 1 Relation between C1-compatibility and ATPG vectors.

回路名	ゲート数	C1-compatibility処理なし			C1-compatibility処理あり			Mb/Ma ×100 [%]	Nb/Na	B/A ×100 [%]
		回路入力数	ATPG ベクトル数	Na・Ma [ビット]	回路入力数	ATPG ベクトル数	Nb・Mb [ビット]			
		Ma	Na	A	Mb	Nb	B			
c432	160	36	27	972	10	33	330	27.8	1.22	34.0
c499	202	41	52	2132	9	52	468	22.0	1.00	22.0
c880	383	60	16	960	14	32	448	23.3	2.00	46.7
c1355	546	41	84	3444	10	84	840	24.4	1.00	24.4
c1908	880	33	106	3498	12	109	1308	36.4	1.03	37.4
c2670	1193	233	45	10485	22	77	1694	9.4	1.71	16.2
c3540	1669	50	86	4300	17	91	1547	34.0	1.06	36.0
c5315	2307	178	42	7476	16	70	1120	9.0	1.67	15.0
c6288	2416	32	11	352	6	18	108	18.8	1.64	30.7
c7552	3512	207	75	15525	26	123	3198	12.6	1.64	20.6
s5378	2779	214	101	21614	22	152	3344	10.3	1.50	15.5
s9234	5597	247	111	27417	30	184	5520	12.1	1.66	20.1
s13207	7951	700	233	163100	31	241	7471	4.4	1.03	4.6
s15850	9772	611	130	79430	35	235	8225	5.7	1.81	10.4
s35932	16065	1763	14	24682	7	16	112	0.4	1.14	0.5
							平均	16.7	1.41	22.2

表 2 ROM テストデータ生成実験結果
Table 2 Experimental results of ROM test data generation.

回路名	ゲート数	Na・Ma［ビット］	Nb・Mb［ビット］	ATPGベクトル単位処理後［ビット］	ビット単位処理後	D/A ×100［%］	D/B ×100［%］	O.H.［%］	CPUタイム［秒］
		A	B	C	D				
c432	160	972 (27)	330 (33)	120 (12)	106	10.9	32.1	11.7	1
c499	202	2132 (52)	468 (52)	180 (20)	153	7.2	32.7	13.4	1
c880	383	960 (16)	448 (32)	168 (12)	151	15.7	33.7	7.0	1
c1355	546	3444 (84)	840 (84)	320 (32)	299	8.7	35.6	9.7	2
c1908	880	3498 (106)	1308 (109)	624 (52)	569	16.3	43.5	11.4	4
c2670	1193	10485 (45)	1694 (77)	1034 (47)	995	9.5	58.7	14.8	8
c3540	1669	4300 (86)	1547 (91)	765 (45)	665	15.5	43.0	7.1	10
c5315	2307	7476 (42)	1120 (70)	496 (31)	442	5.9	39.5	3.4	10
c6288	2416	352 (11)	108 (18)	36 (6)	33	9.4	30.6	0.2	10
c7552	3512	15525 (75)	3198 (123)	1846 (71)	1612	10.4	50.4	8.1	46
s5378	2779	21614 (101)	3344 (152)	1628 (74)	1465	6.8	43.8	9.3	38
s9234	5597	27417 (111)	5520 (184)	3750 (125)	3243	11.8	58.8	10.3	229
s13207	7951	163100 (233)	7471 (241)	4092 (132)	3153	1.9	42.2	7.0	390
s15850	9772	79430 (130)	8225 (235)	5180 (148)	4094	5.2	49.8	7.4	944
s35932	16065	24682 (14)	112 (16)	49 (7)	42	0.2	37.5	0.05	78
		( )内はATPGベクトル数			平均	9.0	42.1	8.1	

れらの数値は実装条件によって変わるため，ここでは単純に CUT 内の論理ゲート数を用いた．この指標 O.H. を平均で約 8%，最悪でも 15%以内に抑えることができ，提案手法が小さな面積オーバヘッドで済むことを示した．とくに，*rpr* 故障を多く含むため面積オーバヘッドを小さく抑えることが困難であると思われていた c2670 や c7552 についても良好な結果を得ることができた．処理時間の項には，ROM テストデータを作成する部分の処理時間，すなわち 3 章の処理手順 3 および 4 に要した時間を載せた．使用計算機は PentiumIII 850 MHz の PC で，OS は Free BSD である．この結果から，本手法は比較的大規模な回路に対しても実用的な時間で処理できていることが分かる．

5. おわりに

本論文では，ROM に格納されたテストデータを 1 ビットずつシフトして検査対象組合せ回路（CUT）の入力パターンとする deterministic BIST 方式に対して，コンパクトな ROM テストデータを生成する一手法を提案した．提案手法では，本来カウンタベース BIST 用に提案された C1-compatibility 処理を行った後に生成された ATPG ベクトル集合の中から少ない ATPG ベクトル数でテストデータを作る手法を提案するとともに，ビット単位でテストデータを圧縮する手法を提案した．また，ISCAS ベンチマーク回路での評価実験により，本手法がコンパクトなテストデータ生成にきわめて有効であることを示した．とくに，c2670 や c7552 等の *rpr* 故障を多く含む回路に対しても少ない面積オーバヘッドのテストデータを生成できることを示した．

参考文献

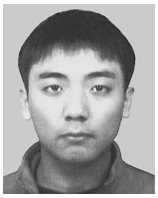
1) Hellebrand, S., Reeb, B. and Tarnick, S.: Pattern Generation for a Deterministic BIST Scheme, *Proc. ICCAD-95*, pp.88–94 (1995).
2) 中尾教伸，小林誠治，畠山一実，飯島一彦，寺田

聖二：BIST 向け検査点挿入方式における遅延・面積オーバーヘッドの低減，電子情報通信学会論文誌 D-I，Vol.J82-D-I, No.7, pp.852–860 (1999).
3) 浅川　毅，岩崎一彦：ATPG ベクトルを利用した BIST テストパターン発生回路，電子情報通信学会論文誌 D-I，Vol.J83-D-I, No.3, pp.360–367 (2000).
4) 市野憲一，斎藤貴之，浅川　毅，福本　聡，岩崎一彦：BIST 指向 n 検出 TPG の提案，電子情報通信学会技術研究報告，Vol.100, No.475, pp.233–238 (2000).
5) Chen, C.A. and Gupta, S.K.: Efficient BIST TPG Design and Test Set Compaction via Input Reduction, *IEEE Trans. CAD*, Vol.17, No.8, pp.692–705 (1998).
6) Hamzaoglu, I. and Patel, J.H.: Reducing Test Application Time for Built-in-Self-Test Test Pattern Generators, *Proc. 18th IEEE VLSI Test Symposium*, pp.369–375 (2000).
7) Gizdarski, E. and Fujiwara, H.: Fault Set Partition for Efficient Width Compression, 電子情報通信学会技術研究報告，Vol.100, No.620, pp.17–24 (2001).
8) Kajihara, S., Pomeranz, I., Kinoshita, K. and Reddy, S.M.: Cost-Effective Generation of Minimal Test Sets for Stuck-at Faults in Combinational Logic Circuits, *Proc. 30th ACM/IEE Design Automation Conf.*, pp.102–106 (1993).
9) Hamzaoglu, I. and Patel, J.H.: Test Set Compaction Algorithms for Combinational Circuits, *Proc. ICCAD-98*, pp.283–289 (1998).
10) Brglez, F. and Fujiwara, H.: A Neural Netlist of 10 Combinational Benchmark Circuits, *Proc. Int. Symp. on Circuits and Systems* (1985).
11) Brglez, F., Bryan, D. and Kozminski, K.: Combinational Profile of Sequential Benchmark Circuits, *Proc. Int. Symp. on Circuits and Systems*, pp.1929–1934 (1989).

(平成 13 年 9 月 13 日受付)
(平成 14 年 3 月 14 日採録)

林　照峯（正会員）

1947 年生．1969 年名古屋大学工学部電気学科卒業．1971 年同大学院工学研究科修士課程修了．同年株式会社日立製作所日立研究所入社．1993 年三重大学工学部電気電子工学科教授．工学博士．LSI の設計自動化，論理回路のテストおよびテスト容易化設計等の研究に従事．電子情報通信学会，IEEE 各会員．



鈴木　孝宣

1978 年生．2001 年三重大学工学部電気電子工学科卒業．現在，同大学院工学研究科修士課程在学中．LSI テストの研究に従事．

戸田　隆宏

1977 年生．1999 年三重大学工学部電気電子工学科卒業．2001 年同大学院工学研究科電気電子工学専攻博士前期課程修了．同年富士通コンピュータテクノロジ株式会社入社．LSI 開発用 CAD 開発業務に従事．

篠木　　剛（正会員）

1954 年生．1977 年東京工業大学理学部情報科学科卒業．1979 年同大学院理工学研究科修士課程修了．同年株式会社富士通研究所入社．1985 年より 1 年間米国オレゴン大学客員研究員．1998 年三重大学大学院工学研究科博士後期課程修了．工学博士．1998 年三重大学工学部講師．1999 年助教授．LSI のテストや設計支援，並列/分散計算機システム，計算機アーキテクチャ，画像処理等に興味を持つ．電子情報通信学会，IEEE 各会員．