

テストパターン変形に基づく  
LSI テストデータ圧縮手法に関する研究

平成 18 年 度

三重大学大学院工学研究科  
博士前期課程 電気電子工学専攻

山 下 智 之

修士論文

テストパターン変形に基づく  
LSI テストデータ圧縮手法に関する研究



平成 18 年度修了

三重大学大学院工学研究科

博士前期課程 電気電子工学専攻

山下 智之

# 目次

<b>1. はじめに</b>	<b>1</b>
<b>2. LSI テスト</b>	<b>4</b>
2.1 LSI テストの必要性	4
2.2 LSI テストの概要	5
2.3 論理回路の故障検出	6
2.4 ドントケア	7
<b>3. テスト容易化設計</b>	<b>9</b>
3.1 テスト容易化設計の必要性	9
3.2 スキャン方式	9
<b>4. テストデータ圧縮</b>	<b>12</b>
4.1 LSI テストにおけるテストデータ圧縮の必要性	12
4.2 テストデータの圧縮・解凍	12
4.3 圧縮後のテストデータ	13
<b>5. 提案手法</b>	<b>16</b>
5.1 SDI 法とその問題点	16
5.2 テストパターン変形	19

<b>6. 実験と評価</b>	<b>25</b>
6.1 実験対象	25
6.2 単語の種類数の評価実験	26
6.3 テストデータ量・圧縮率の評価実験	28
6.4 他のテストデータ圧縮手法との比較	30
<b>7. まとめ</b>	<b>32</b>
<b>謝辞</b>	<b>33</b>
<b>参考文献</b>	<b>34</b>
<b>発表論文</b>	<b>36</b>

# 1. はじめに

現在，LSI は身近な家電製品，医療製品にまでさまざまな分野で使用されている．そのため，LSI の製造後に故障がないかどうか検査する製造検査が重要となっている．しかし，近年の利用分野の拡大に伴う LSI の製造量の増加により，その製品検査であるテストのコスト増加が問題となっている．そのため，このテストコストの削減が重要となってきた．テストコストには，テスト装置（LSI テスタ）のメモリ容量に関わるテストデータ量，LSI テスタの効率的利用に関わるテスト時間，テストの信頼性に関わるテスト時消費電力などがある．これらのどれが最も重要であるかは LSI の特性やテスト環境等によって変わるので，状況に応じたテストコスト削減が必要である．本研究では，それらのなかでも，テストデータ量の削減が最も本質的であると考え，テストデータ量の削減を目的とする．

回路の大規模化，複雑化に伴ってテストデータ量は多くなる．そのため近年，テストデータを格納するテスト装置のメモリ容量不足が問題となってきた．このメモリ容量不足を解消するための方法のひとつとして，メモリを増設する方法がある．しかし，LSI テスタには高速，かつ，高信頼の特殊なメモリが必要であるため，その増設には限界がある．そこで，テスト品質を落とさずにテストデータ量を削減し，必要とするメモリ容量を削減する必要がある．これを実現するためのテストデータ量の削減法が，従来より多く研究されてきた [1～8]．

テストデータ量の削減には，大きく分けて 2 種類の方法（テストコンパクション，テスト圧縮）がある．テストコンパクション（test compaction）[1,2] とは，故障検出に必要なないテストパターンを除くことによりテストデータ量を削減することである．スキャン方式を前提とするテストでのテストデータ量の

削減には、テストコンパクションが有効であることが、よく知られている。また、テスト圧縮 (test compression) [3~8] はテストデータの情報量を減らすずに、データの表現方法を工夫して記憶しておくべきテストデータ量を減らすものである。テスト圧縮されたテストデータは、被検査 LSI に入力され、LSI 内の解凍回路を介して元のテストデータに解凍 (一般の圧縮では、「展開」「伸長」と呼ばれることが多いが、ここでは「解凍」と呼ぶ) され被検査回路 (CUT: circuit under test) に入力される。

テストコンパクションについては、さまざまな研究の結果、優れた手法がすでに実現されている。したがって、さらにテストデータ量を削減するためには、テストコンパクションとテスト圧縮を併用する方法が効果的であると考えられる。そこで本論文では、テストコンパクションしたテストデータを対象にテスト圧縮を行う。

テスト圧縮を行う方法では、解凍時に圧縮前と圧縮後の対応関係を表した辞書が必要である。そのため、辞書も LSI テスタから解凍回路に入力する必要がある。したがって、テスト圧縮に際してこの辞書のデータ量も考慮する必要がある。

辞書のデータ量を考慮した方法の一つに選択的ドントケア検出に基づく圧縮手法 (SDI 法) [9] がある。この方法では、ドントケア検出を繰り返し、X 値 (ドントケアビット: 0, 1 どちらでも良いビット) をうまく 0, 1 値 (ここでは, 0, 1 の値をケアビットと呼ぶ) に固定することによりテストデータそのものの圧縮のみでなく、辞書のデータ量を減らすことも行っている。しかし、コンパクションしたテストデータでは、ドントケアの割合が小さいため SDI 法では辞書のデータ量の削減に限界がある。そこで、本論文では、ドントケアビットのみでなくケアビットについても故障検出率低下を起こさないように値を変えること

により，さらなる辞書のデータ量を削減することを試みる．また，テストパターンを増やすことを許容することで，ドントケアビットを増やし，さらにケアビットの値も変えやすくする．これを実現する方法としてテストパターン変形を提案する．本研究で提案する手法は，標準的な多重スキャン方式におけるテストデータを対象としテストデータ圧縮を行うことで，テストデータ量の問題を解決し，従来手法で圧縮したときの圧縮率よりもさらに圧縮率を向上させることを目的とする．

本論文は以下のように構成する．第 2 章では LSI テストの概要を説明する．第 3 章では本論文で扱うテストデータの圧縮と解凍について述べる．第 4 章では提案するテストデータ圧縮法について述べる．第 5 章で提案手法を用いた実験を行い，評価をする．最後に，第 6 章でまとめる．

## 2. LSI テスト

回路になんらかの物理的欠陥があった場合，その回路は正しい動作をしなくなる，すなわち故障する可能性がある．LSI テストは，この故障を検出するために行われる検査である．工場で製造された LSI は出荷前に検査され，故障が検出されたものは出荷されない．もし，故障がある LSI がそのまま出荷されてしまうと，LSI の誤作動によって，大きな問題を起こす可能性がある．そのため，正確に LSI テストを行うための技術が必要である．本章では，LSI テストに関する基本的な概念の説明を行う．

### 2.1 LSI テストの必要性

LSI のテストは，本来，設計や製造が完全ならば不要である．しかし，半導体技術の微細化により現在の LSI の製造は原子のオーダーで行われるようになっていたため，一つのチップ内のトランジスタでさえも，すべて同じ特性で動作するとは言いがたくなってきている．このため，あらゆる条件下での性能を保証する設計の実現は困難となっているため，不良品の発生は避けられない．また設計が完全であったとしても，LSI は次々に高性能化しており，時代の先端を走る LSI 製品ほどプロセスが成熟し安定する前に量産を開始する．そのため，量産の段階で不良な箇所ができてしまい，不良品になってしまうこともある．このような不良品の流出を防ぐのが LSI テストを行う目的である．LSI テストを行うことは LSI の信頼性を保つために非常に重要である [10] ．



## 2.2 LSI テストの概要

LSI テストの概略を図 1 に示す。LSI テストは、LSI 内部の故障を検査するために必要なテスト入力データを LSI テスタが被検査回路（CUT：Circuit Under Test）に入力し、そのときの CUT の出力データを正常時の時の出力データと比較することにより行われる。一般に、テストに必要なテスト入力データとテスト出力データをまとめてテストデータと呼ぶ。

LSI テストには、LSI の大規模化に伴いテストデータ量などのテストコストが増大してしまう問題がある。現在、テストコストをより小さく抑える研究が広く行われており、本研究ではテストデータ量の削減を目的とする。

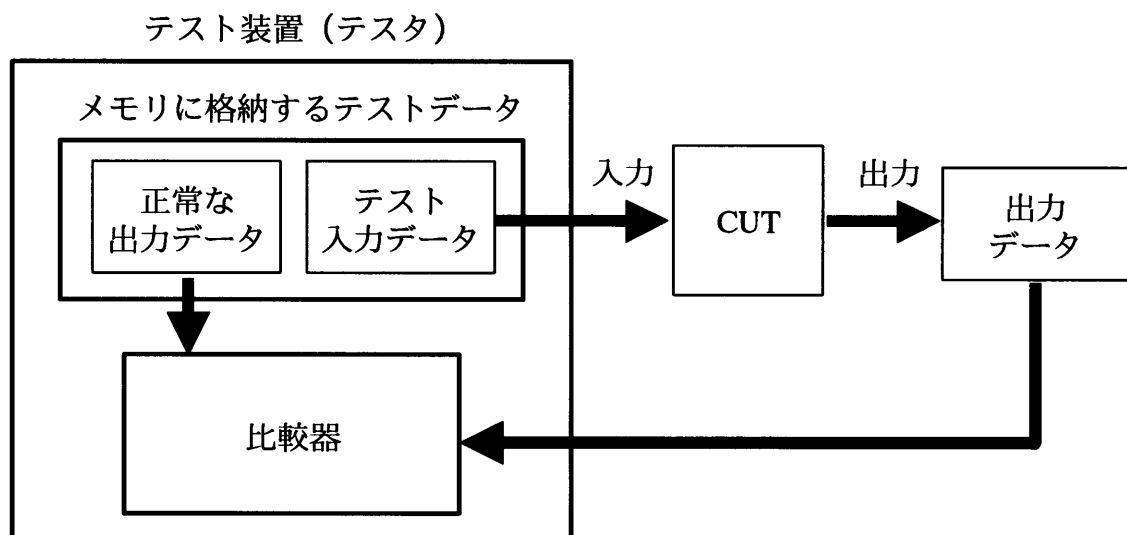


図 1 LSI テスト

## 2.3 論理回路の故障検出

故障とは、回路内の配線に生じる断線や短絡など、物理的に生じた原因により、回路を構成する要素が予め規定された機能を失うことを意味する。論理回路においてこのような故障の一つに、正常な論理値が 1 のところが 0 になったり、正常な論理値が 0 のところが 1 になったりするものがある。このような誤りを起こす故障を論理故障という（以下、単に故障と呼ぶ）。故障には、論理ゲートの入出力線の値が 0 または 1 に固定される縮退故障がある。回路内に故障がただ一つ存在する場合、これを単一故障、複数の故障が存在する場合、多重故障という。

回路の製造時やシステムの保守時に、回路内に故障が存在するかどうかを調べることを故障検出、またはテストという。故障にはさまざまなものがあり、テストについて考える場合、どのような故障を対象とするかを決定する必要がある。通常、故障として信号線の単一縮退故障を仮定することが多い。これは、多くの故障が論理回路の入出力線の単一縮退故障を検出するテストパターン集合によって検出できることが知られているからである [11]。よって、本研究でも、テストデータとして単一縮退故障を検出するテストパターン集合を対象とする。

## 2.4 ドントケア

テストパターンには、その値が 0 でも 1 でもよいドントケアというビットが含まれている。テストコスト削減のために、このドントケアのビットはしばしば利用されている。本研究においてもテストコストを削減するための手段として、ドントケアを利用する。

例えば図 2 の回路において、NAND 素子の出力が 1 に固定してしまう 1 縮退故障を検出する場合を考える。信号値が 1 に縮退している信号線の信号値が 0 となるように NAND 素子に入力を与えることで、この故障は検出できる。この値を得るためには NAND 素子の入力は両方とも 1 が入力されなければならない。これにより NOT 素子の出力が 1 とならなければならないため、NOT 素子に入力される値は 0 となる。そして OR 素子の出力値も 1 としなければならない。このためには 2 つあるうちの入力値のどちらか一方の入力を 1 にすればよい。その結果、もう片方の入力は 0, 1 のどちらでもよい値となる。この 0, 1 どちらでもよい値を X (ドントケア) と呼ぶ [12]。

LSI のテストデータにはこのドントケアが多く含まれている。ここで、あるテストパターン  $p1: 10X$  と  $p2: X0X$  においてこれらは、テストパターンとしては違うものであるが、ドントケアが 0, 1 どちらでもよいということに着目して、0 または 1 に適切に決めることによりこれらのテストパターンは同じパターンとすることができる。このような性質をもつ 2 つのテストパターンはお互いに compatible であるという。この例では、 $p1$  と  $p2$  は compatible である。

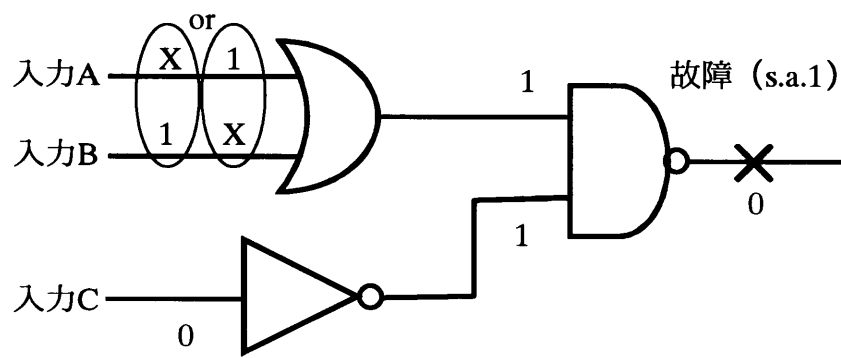


図2 テストデータにドントケアを含む組み合わせ回路

## 3. テスト容易化設計

### 3.1 テスト容易化設計の必要性

LSI の大規模化、複雑化により、そのままでは内部回路のテストを十分に行うことが困難となってきたため、テスト容易性を設計段階から考慮して設計することが重要である。現在では、テスト容易化手法として、種々の手法が実用化されている。一般には、スキャン方式がよく用いられる。

次節で、本研究で対象とするテスト容易化手法であるスキャン方式について詳しく述べる。

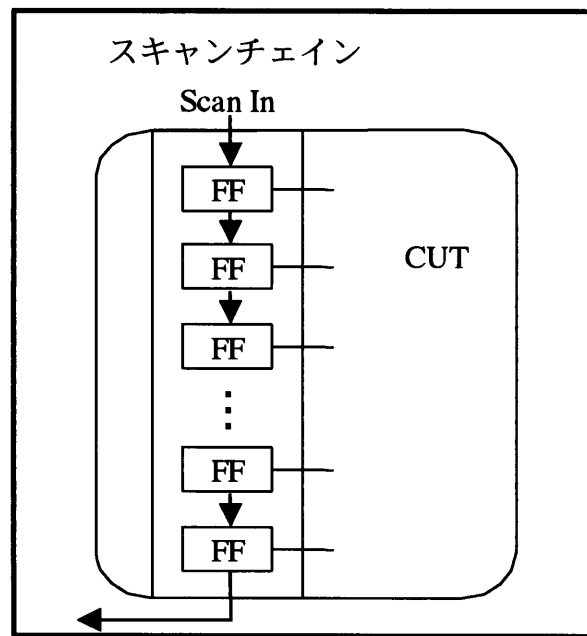
### 3.2 スキャン方式

順序回路に対するテストパターン生成は、組み合わせ回路に対するテストパターン生成と比べて困難である。しかし、LSI は順序回路である。また、近年の LSI の大規模化・複雑化に伴いより困難なものとなっている。そこで、LSI をテストする際に、その回路を組み合わせ回路へと変換することによって、テストパターン生成が容易になる。このため、もとの回路とは異なる回路によってフリップフロップ（以降 FF と略す）の値を容易に設定できるようにする。スキャン方式では、CUT に含まれる FF を、シフトレジスタ状に数珠つなぎにする（スキャンチェーンを構成する）ことで、これを実現する。このとき、FF はスキャン方式に使用できる専用の FF（scan FF）に置き換えたうえで、スキャンチェーンを構成する。この結果、スキャンチェーンの端（SI : Scan In）から 1 ビットずつ値を送り込むことで、各 FF に値を設定できる。

一般にスキャン方式では、順序回路にもともと含まれていた FF だけでなく、回路の各入力端子にも scan FF を設置し、ひとつの SI から、被検査回路の全入力・全 FF にデータを与えるようにスキャンチェーンを構成する（シングルスキ

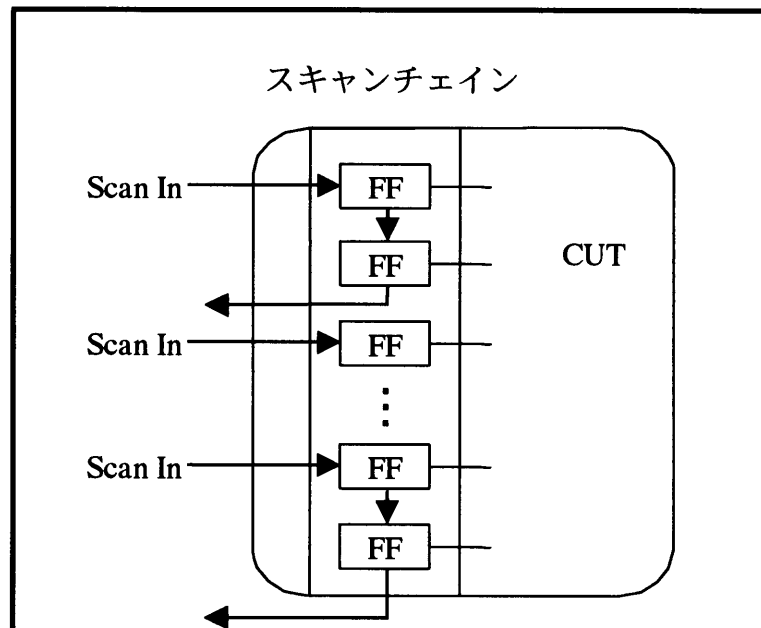
ャン構造)。しかし，被検査回路の大規模化に伴いスキランチェーンが長くなり，すべてのデータを被検査回路へ送り込む時間がかかるようになる．そこで，複数のスキランチェーン（多重スキラン構造）を用いることで，その時間を短縮することも行われている．図 3 に，シングルスキラン構造，多重スキラン構造の概略を示す．本研究では，多重スキラン構造をもつ回路を対象に行う．

LSI



(i) シングルスキャン構造

LSI



(ii) 多重スキャン構造

図3 スキャン構造

## 4. テストデータ圧縮

本章では、一般に行われているテストデータ圧縮について説明する．また，テストデータの圧縮と解凍の説明に使われる記号や用語の説明も行う．

### 4.1 LSI テストにおけるテストデータ圧縮の必要性

LSI テストにおいては，LSI テスタが LSI 内部の故障を検査するために必要なテストデータを LSI に入力し，その応答を観測し，正常に応答したときの観測パターンと比較して故障を検出する．最近では，LSI の大規模化，複雑化に伴い，LSI の故障を検出するためのテストデータ量が増大してきており，このテストデータを記憶しておくテスト装置のメモリ容量不足などの問題を招いている．LSI テスタのメモリは高価なために，安易に増設できない．そこで，テストデータ量の問題を解決する方法として，テストデータを圧縮することで LSI テスタのメモリに記憶するテストデータ量を削減する方法がある．そのため，圧縮率の高い圧縮方法が求められている．

### 4.2 テストデータの圧縮・解凍

テストデータ圧縮は通常，入力側，出力側の両方で行う．出力データの圧縮は MISR (Multiple Input Signature Register) を使った手法がよく用いられる．出力データの圧縮も研究対象の一つとしてあるが，本研究では出力側のテストデータは扱わず，入力側のテストデータの圧縮のみを対象とする．よって，本論文では，入力側のテストデータを単にテストデータと呼ぶ．



## 4.3 圧縮後のテストデータ

### 4.3.1 データ構成

一般にテストデータを圧縮すると、圧縮後のテストデータは、圧縮されたテストデータと解凍時に用いる辞書の二つから構成される。本研究では、圧縮方法として圧縮効率が高いハフマン法を用いる。よって、今回用いるハフマン法を用いた場合を例に説明を行う。ハフマン法を用いたテストデータ圧縮では、最初にテストデータを一定長のビットに区切る。区切った固まりを単語とする。そして、単語の頻度を取り、頻度に基づいて符号を作成し圧縮を行う。圧縮を行うことにより、圧縮後のテストデータとして、圧縮されたテストデータと、符号と単語との対応を表した辞書が作成される（図 4）。また、テストデータの圧縮を行う場合、単語の長さ  $L$  によって、圧縮されたテストデータ量と辞書のデータ量が変わってくる。これについて、次の節で説明をする。

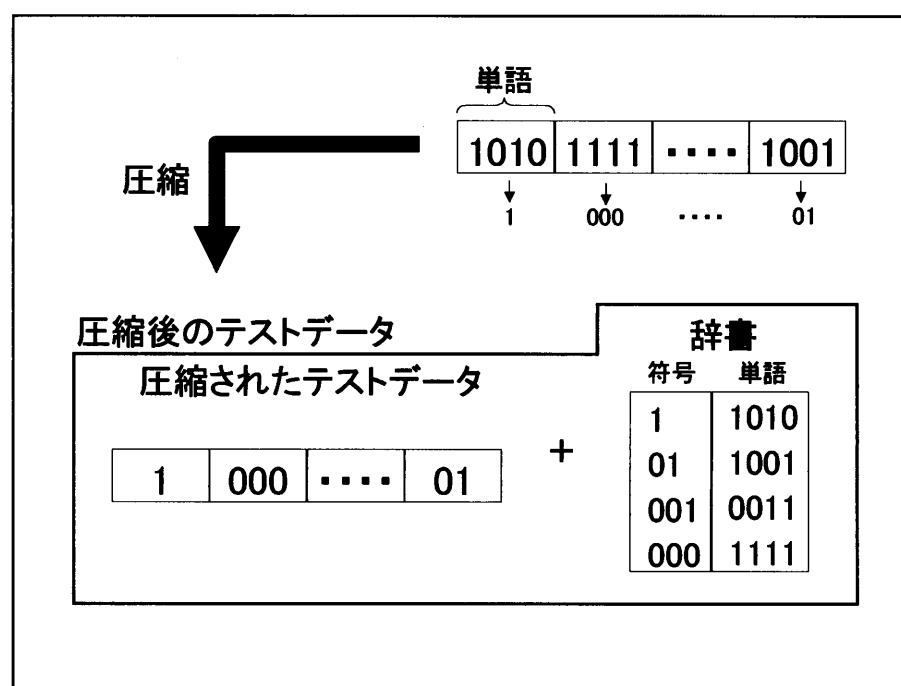


図 4 圧縮後のテストデータ

#### 4.3.2 単語の長さによる構成の違い

単語の長さ  $L$  を 4 ビット, 8 ビットにとった時の辞書を図 5 にそれぞれ示す。テストデータ量は単語の長さ  $L$  を長くとるほど, より長いビットを短いビットに割り当てることになりテストデータ量は小さくなる。逆に, 辞書のデータ量は単語の長さが長くなるほど大きくなる。これは, 単語の長さ  $L$  が長くなったために単語の種類数が多くなってしまうためである。そして, より長い単語を記憶しておかなければならないので辞書のデータ量は大きくなる。このことから, テストデータ圧縮では単語の長さ  $L$  を大きくするほどテストデータ量は小さくできるが, 辞書のデータ量は大きくなってしまう。図 6 に, ハフマン法により圧縮を行う場合の単語の長さ  $L$  を変えたときのテストデータ量と辞書のデータ量の関係を示す。図 6 を見てもわかるように単語の長さ  $L$  が長くなるほどテストデータ量は小さくなり, 辞書のデータ量は大きくなっている。

辞書のデータ量の大きさを決める最大の要因は単語の種類数である。したがって, 単語の長さ  $L$  を大きくとっても, 単語の種類数を少なくすれば, 辞書のデータ量を小さくでき, 総テストデータ量は小さくなる。

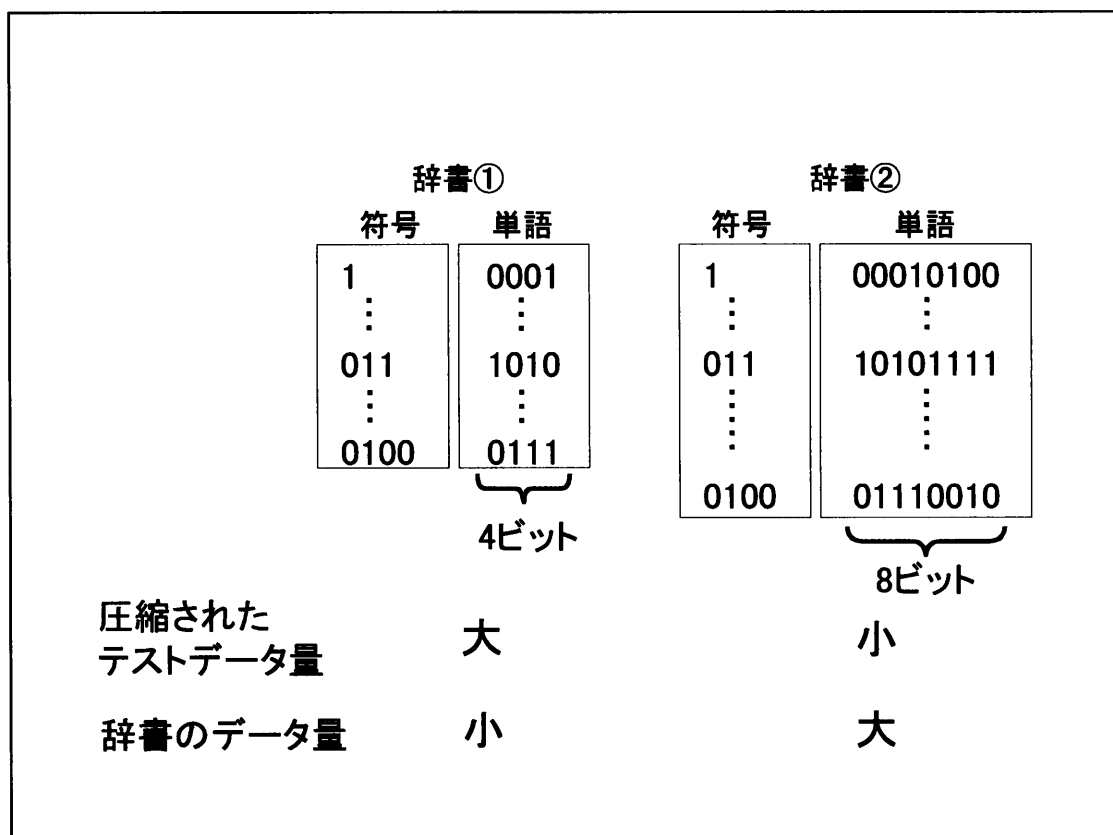


図5 辞書

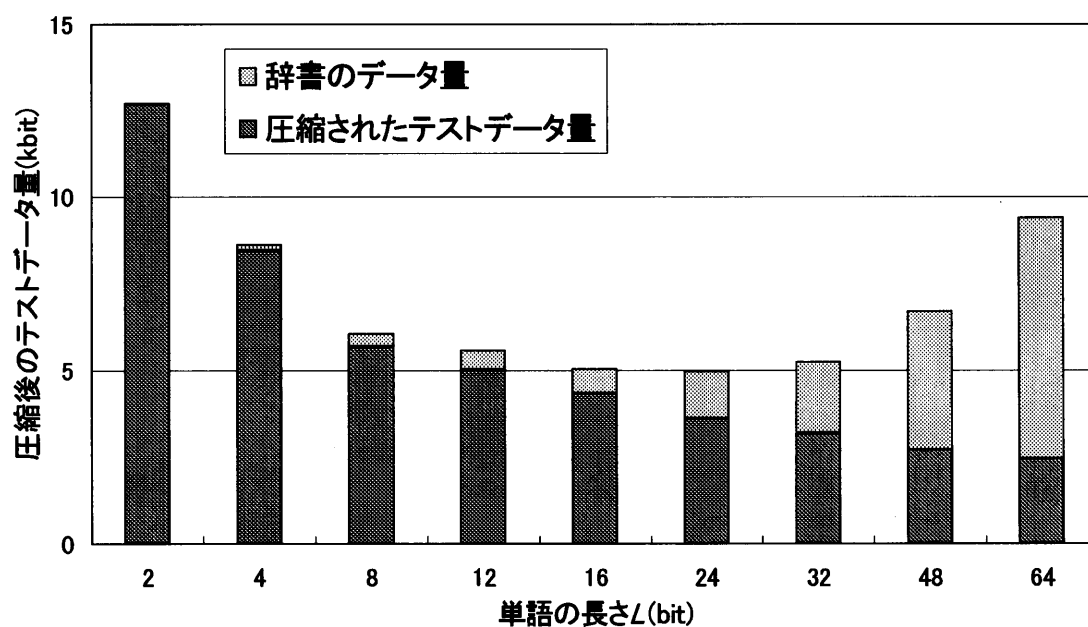


図6 圧縮後のテストデータ量 (s5378)

## 5. 提案手法

本研究では，テスト圧縮において単語の長さ  $L$  を大きくとったときの辞書のデータ量を少なくすることにより，LSI テスタに格納する総テストデータ量を少なくする．辞書のデータ量が大きくなる原因は，単語の種類数の増加である．したがって，単語の長さ  $L$  を大きくとっても，単語の種類数を少なくすれば，辞書のデータ量を小さくでき，総テストデータ量は小さくなる．そこで，テストデータが一般のデータとは違い，故障検出率が低下しなければデータを変えても良いということを利用し，圧縮前の辞書に格納する単語の種類数を少なくすることでデータ量を少なくする．同じように，辞書に記憶する単語の種類数を少なくすることによりデータ量を少なくする方法として選択的ドントケア検出に基づく圧縮手法（SDI 法）[9] がある．次の節で SDI 法の圧縮方法と問題点について説明する．

### 5.1 SDI 法とその問題点

#### 5.1.1 SDI 法の圧縮方法

SDI 法は，テストデータ中に含まれるドントケアビットをうまくケアビットにすることによって単語の種類数が少なくする方法である．そして，圧縮中にも新たにドントケアになる部分を見つけることにより単語の種類数削減の効果をあげている．SDI 法における単語の種類数削減処理では，次の 2 つの処理を繰り返す．

- (1) compatible な単語を一つに併合することを繰り返すことによって単語の集合を作る．
- (2) 出現頻度の大きい単語については，X 値を 2 値固定し，出現頻度が小さい単語についてはドントケア検出を行い，X 値をもつ部分を増やす．

SDI 法では、圧縮中もドントケア検出を繰り返すことで、圧縮率を向上させている。すなわち、(2) の操作において、一度に全ての  $X$  値を 2 値固定してしまうのではなく、出現頻度の大きい単語だけを 2 値固定し、出現頻度の小さい単語についてはドントケア検出を行っている。これは、一部を 2 値固定した結果、前のドントケア検出ではドントケアにならなかった部分に、新たに  $X$  値が現れる可能性があるためである。なお、この処理は、出現頻度が大きいとみなされる単語の数を徐々に増やしながら行われる。また、2 値固定は単語の種類数が減少するように行っている。

#### 5.1.2 SDI 法の問題点

SDI 法での単語の種類数削減効果は、テストデータに含まれるドントケアの割合に大きく依存する。しかし、一般に用いられるテストデータ（コンパクションしたテストデータ）では、ドントケアの割合が小さいため、どうしても出現頻度の小さい単語が多く残ってしまい辞書に格納する単語の種類数の削減に限界がある。例を図 7 に示す。図 7 では、SDI 法によって、あるテストパターンに含まれる単語 0110 をなくすことを考える。この単語を含むテストパターンでしか検出できない故障（必須故障）が  $f_1 \sim f_3$  の 3 個あるとする。この場合、 $f_1 \sim f_3$  を検出できなければいけないので、この単語のどのビットもドントケアにすることができない。そのため、この単語 0110 をなくすことはできない。このように、ドントケアにできないビットを多くもつ単語は、ドントケア検出の効果があまり期待できないため、単語の種類数を削減できないことがある。また、単語の長さ  $L$  が大きくなるほど単語内のケアビットが多くなるため、単語の種類数を少なくできない。

そこで、本研究では、SDI 法を改良することにより、より単語の種類数を削減し、単語の長さ  $L$  が大きいときにも有効に働くようにする。SDI 法での問題点の

解決法として、ドントケアビットだけでなくケアビットの値も、故障検出の能力を低下させない範囲で変化させることで、さらに単語の種類数を減らす手法を提案する。テストパターンを増やすことを許容することで、ケアビットの制約を緩くし、単語の種類数削減をしやすくする。

テストパターン …0110…		単語
		0100
		1110
		1000
		0000
		0110
必須故障	テストパターン	
f1	: …0xx0…	
f2	: …01xx…	
f3	: …x110…	

図7 SDI法の単語の種類数削減の限界

## 5.2 テストパターン変形

### 5.2.1 基本的な考え

この方法は、あるテストパターンだけで検出する必須故障を検出し続けるように、そのテストパターンを変化させて、そのテストパターンに含まれる出現頻度の小さい単語をなくすものである。また、新しい二つのテストパターンと置き換えることにより、必須故障を二つのテストパターンに分けることも許容する。これにより、出現頻度の小さい単語をなくしやすくする。

(テストパターン変形では、テストパターンを 3 以上のテストパターンに置き換える方法も考えられる。しかし、本研究では、圧縮にかかる時間とテストパターン数が大きく増えてしまうと考え、3 以上に置き換える方法は行わない。)

例を図 8 に示す。図 8 では、図 7 と同様に単語 0110 をなくすことを考える。そこで、提案する方法では、一つのテストパターンを二つのテストパターンにし、単語 0110 を 01x0 と x110 の 2 つの単語に分けることを考える。これにより、単語 01x0 では、必須故障  $f_1$  と  $f_2$  が検出でき、単語 x110 では、必須故障  $f_3$  が検出できる。このように、故障検出率が低下しないように 2 つの単語に分けることにより単語 01x0 は単語 0100 にすることができ、単語 x110 は単語 1110 にすることができる。その結果、単語 0110 をなくすことができる。これにより、単語の種類数の削減ができる。この方法では、単語の長さ  $L$  が大きい場合でも、テストパターンを二つに分けることによってドントケアビットの割合を多くすることができるので単語の種類数の削減に有効であると考えられる。これを実現する方法としてテストパターン変形手法を提案する。なお、テストパターンを二つにした結果、テストパターン数が増えるが、単語の種類数の削減効果により圧縮後のテストデータ量は減少すると予想される。

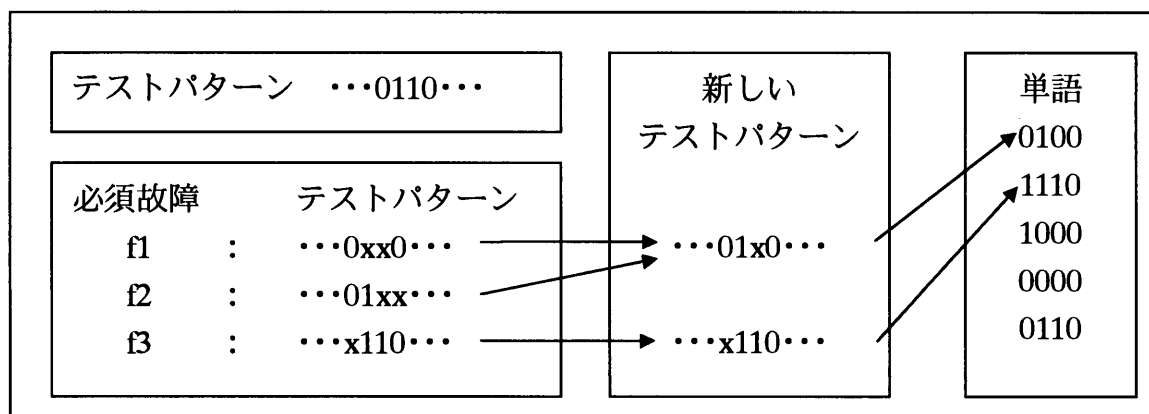


図8 提案手法の単語の種類数削減

### 5.2.2 テストパターン変形の方法

テストパターン変形では、単語の種類数を少なくする方法として出現頻度の小さい単語を含むテストパターンを出現頻度の大きい単語のみで構成された新しいテストパターンに置き換える。また、一つのテストパターンを二つのテストパターンで置き換えることも用いることにより単語の種類数を少なくしやすくする。また、置き換えを行う条件として故障検出率が低下しないものとする。本研究では、置き換えを行う時に元のテストパターンでしか検出できない故障（必須故障）をすべて検出できる新しいテストパターンに置き換えることにより故障検出率が低下しないようにする。この方法では、テストパターン数が増えてしまうため、圧縮されたテストデータのデータ量が少し大きくなる。しかし、大幅に単語の種類数を削減できるので、辞書のデータ量を大きく減らすことができ、結果として、LSI テスタに格納するデータ量を削減することができる。テストパターン変形の手順を以下に示す。



1. テストデータの中で出現頻度が一番小さい単語を含むテストパターンを一つ選ぶ。(図9(a)では $t_1$ となる.)
2. 故障シミュレータでこのテストパターンでしか検出できない故障(必須故障)を調べる。(  $t_1$  の必須故障は図9(b)の  $f_1 \sim f_5$  となる.)
3. テストパターン中の出現頻度の小さい単語を出現頻度が一番大きい単語で置き換えたテストパターンを作成する。(図9(b)では, 単語 0000 で置き換えたテストパターン  $t_1'$  となる.)
4. このテストパターンで必須故障をすべて検出できるかを調べる.
5. 全ての必須故障を検出できれば, 元のテストパターンを新しいテストパターンと置き換える. 手順1に戻る.(図9(b)では,  $t_1'$  ですべての必須故障が検出できるので置き換えを行う. 置き換えることにより単語 0011 をなくすことができる.)
6. 検出できない必須故障がある場合, 次に出現頻度の大きい単語に置き換え手順4~6を繰り返す.
7. 以上の結果, 置き換えができない場合, 手順8~11(図9(c))を行う.  
(手順4~6ができなかったものとして  $t_4$  を説明する.)
8. 出現頻度の小さい単語を出現頻度の大きい単語のどれかで置き換えたテストパターンのなかで, 必須故障を一番多く検出できるものを選択する.  
(図9(c)では, 単語 1001 で置き換えたテストパターン  $t_4'$  となる. このテストパターンでは必須故障  $f_6, f_7, f_9, f_{10}$  が検出できる.)
9. 出現頻度の大きい単語のどれかで置き換えたテストパターンのなかで, 手順8で選択したテストパターンでは検出できなかった残りの必須故障をすべて検出でき, その中で一番出現頻度の大きい単語で置き換えたテストパターンを選択する.(図9(c)では, 単語 0000 で置き換えたテストパ

ターン $t_4$ ”となる．このテストパターンでは必須故障  $f_6, f_8, f_{10}$  が検出できる．)

10. 全ての必須故障を検出できる二つのテストパターンがあるならば，元のテストパターンをこれら二つのテストパターンと置き換える．（図 9 (c) では，これら二つのテストパターンで全ての必須故障を検出できるので，元のテストパターンと新しい二つのテストパターンと置き換える．)
11. 残りの必須故障をすべて検出できる二つ目のテストパターンがない場合は，この単語の置き換えを行わない．次に出現頻度の小さい単語を含むテストパターンにおいて，手順 2～11 を繰り返す．

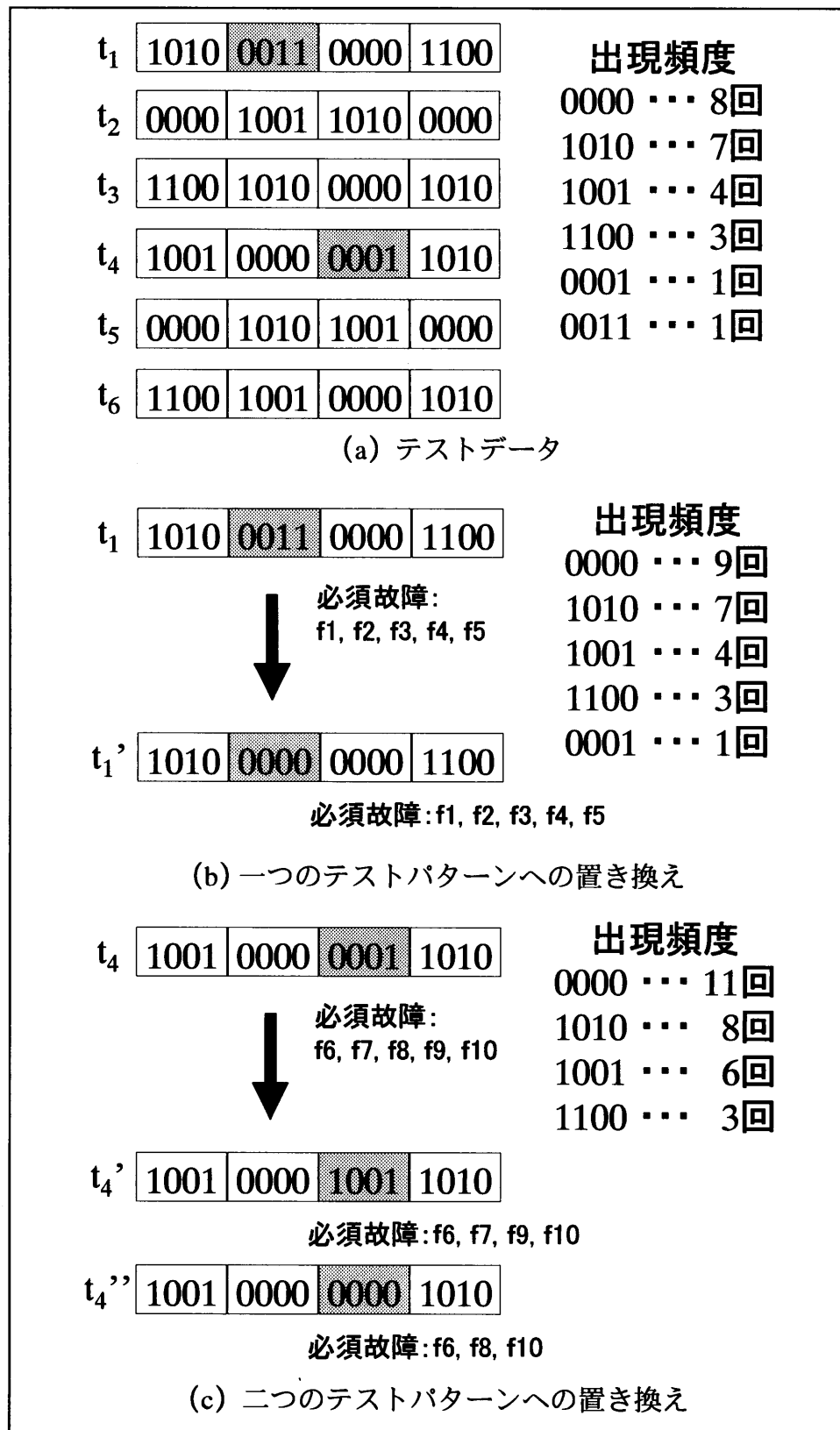


図9 テストパターン変形

### 5.2.3 提案する圧縮手法

本論文では,SDI 法に提案するテストパターン変形を付け加えたテスト圧縮手法を用いる. SDI 法の処理 (1) と (3) の間に提案するテストパターン変形 (2) を加えることにより,さらなる単語の種類数の削減を図るテスト圧縮手法である. 提案する圧縮手法はこの (1) ~ (3) の処理を繰り返して行う. また,テストパターン変形(2)の処理は,出現頻度の小さいとみなされる単語を徐々に増やしながら行う.

- (1) compatible な単語を一つに併合することを繰り返すことによって単語の集合を作る.
- (2) 4.2.2 で述べたテストパターン変形を行う.
- (3) 出現頻度の大きい単語については,ドントケアビットを2値固定する. 出現頻度の小さい単語についてはドントケア検出を行い,X 値をもつ部分を増やす.

## 6. 実験と評価

提案手法の有効性を確認するために，提案手法をプログラムとして実現し，PC 上で実験を行った．

### 6.1 実験対象

今回の実験において，辞書に格納する単語の種類数，テスト時間は被検査回路に入力するテストパターン数で評価した．実験の比較対象はハフマン法のみで圧縮した方法（Huffman coding），SDI 法にハフマン法を適応した方法（SDI + Huffman coding），そして今回提案する手法（Proposed）である．

実験には ISCAS'89 ベンチマーク回路を，フルスキャン回路付きであるものとして用いた．対象回路として s5378, s9234, s13207, s15850, s35932, s38417, s38584 の 7 つの回路を用いた．各回路に対して，故障検出効率 100% の X 値を含むコンパクトテスト集合を用いた．

## 6.2 単語の種類数の評価実験

### 6.2.1 単語の種類数の比較目的

ここでは、各手法を用いたときの辞書に格納する単語の種類数を比較する。本論文で提案する手法は、辞書に格納する単語の種類数を減らすことによって辞書のデータ量を少なくすることを目的とした手法である。そこで、その効果を評価するために、この実験を行う。

### 6.2.2 単語の種類数の評価

各手法（Huffman coding, SDI + Huffman coding, Proposed）でテスト圧縮を行った場合の辞書に格納する単語の種類数の結果を表 1 に示す。また、テストパターン数の増加における単語の種類数の削減効果をみるために、テストパターン数を増加させない場合（※1）の結果も表 1 に示す。それと、提案手法の効果を表すために、SDI + Huffman coding（表 1 の 4 列目）の結果と提案手法（表 1 の 6 列目）の比率を示した。単語の長さ  $L$  は各回路ごとに同じ長さである。

テストパターン数を増加させない場合では、SDI + Huffman coding よりも単語の種類数をすこし少なくすることができた。Proposed では、SDI + Huffman coding よりも大幅に単語の種類数を少なくすることができた。また、提案手法の方が単語の種類数が小さいもので 0.08 倍、大きいもので 0.27 倍、平均では 0.17 倍に小さくできた。このことから、テストパターン数の増加が単語の種類数削減に大きな効果があるとわかり、提案手法が単語の種類数削減において有効であると考えられる。

表 1 辞書に格納する単語の種類数の比較

Circuit	L	Huffman coding	SDI + Huffman coding (A)	※1	Proposed (B)	Ratio B/A
S5378	54	149	113	100	23	0.20
S9234	62	190	154	119	41	0.27
s13207	88	107	62	56	16	0.26
s15850	88	190	135	121	27	0.20
s35932	126	103	95	94	8	0.08
s38417	128	420	348	346	27	0.08
s38584	122	363	258	244	30	0.12
average	-	-	-	-	-	0.17

※1：Proposed においてテストパターン数を増加させない場合

## 6.3 テストデータ量・圧縮率の評価実験

### 6.3.1 提案手法を用いたテスト圧縮

ここでは、テストパターン変形によってテストパターン数が増えることによる圧縮されたテストデータ量の増加と単語の種類数の削減における辞書のデータ量の減少を評価する。これにより、提案手法のデータ量削減の効果をみる。

### 6.3.2 提案手法を用いたテスト圧縮結果

提案手法(Proposed)でテスト圧縮を行った場合の圧縮後のテストデータ量(圧縮されたテストデータ量, 辞書データ量)と圧縮率の結果を表 2 に示す。単語の長さ  $L$  は, 各回路ごとに回路規模を考慮した, ある程度大きい長さをとった。ここで, 圧縮率は  $CR=1-(T_E+V_C)/T_D$  で計算した。結果として, 圧縮率が高いもので 94.7%, 低いもので 75.4%, 平均で 84.8% となった。このことから, 提案手法ではテストパターン数が増えてしまうが, 5.2 節で述べたように大幅に単語の種類数を削減することができ, 結果として圧縮後のテストデータ量を少なくすることができた。これにより, 提案手法によるデータ量削減が有効であるとわかった。



表 2 提案手法を用いたテスト圧縮の結果

Circuit	$N_T$	$T_D$	$N_T$ After	L	$T_E$	$V_C$	$T_E+V_C$	CR(%)
s5378	99	21186	194	54	2822	1469	4291	79.7
s9234	110	27170	224	62	3684	2988	6672	75.4
s13207	233	163100	324	88	7113	1570	8683	94.7
s15850	97	59267	298	88	6250	2654	8904	85.0
s35932	12	21156	63	126	1515	1067	2582	87.8
s38417	86	143104	516	128	22037	3751	25788	82.0
s38584	111	162504	468	122	14026	3986	18012	88.9
average	-	-	-	-	-	-	-	84.8

$N_T$  : 圧縮前のテストパターン数

$T_D$  : 圧縮前のテストデータ量

$N_T$  After : 圧縮後のテストパターン数

L : 単語の長さ

$T_E$  : 圧縮されたテストデータ量

$V_C$  : 辞書のデータ量

$T_E+V_C$  : 圧縮後のテストデータ量

## 6.4 他のテストデータ圧縮手法との比較

### 6.4.1 他のテストデータ圧縮手法との比較目的

ここでは、他のテストデータ圧縮手法と比較することで、提案手法の有効性を評価する。提案手法を用いたテスト圧縮の結果を表 2 に示す。提案手法の圧縮後のテストデータ量と、従来手法で圧縮したときの圧縮後のテストデータ量を表 3 に示す。なお、従来手法として SDI 法だけでなく、Mintest [2], Bay ら [4], Chan ら [5], Li ら [8], Hayashi ら [14] の手法も含めた。

### 6.4.2 他のテストデータ圧縮手法との比較結果

圧縮後のテストデータ量について従来手法と提案手法の結果を表 3 に示す。単語の種類数削減からのテストデータ量の削減を直接比較するため、SDI + Huffman coding (表 3 の 7 列目) の結果と提案手法 (表 3 の 8 列目) による結果を載せるとともに、比率を示した。その結果、提案手法の方が圧縮後のテストデータ量が小さいもので 0.32 倍、大きいもので 0.86 倍、平均で 0.69 倍と全体的にテストデータ量を少なくすることができた。また、他の従来手法と比較しても一部の回路を除き、圧縮後のテストデータ量を少なくすることができた。

表 3 圧縮後のテストデータ量の比較

Circuit	$T_E$				$T_E+V_C$			Volume ratio B/A
	Mintest '98[2]	by Bay. '01[4]	by Chan. '02[5]	by Li '03[8]	by Hayashi '04[14]	SDI + Huffman coding (A)	Proposed (B)	
s5378	20,758	-	-	6,345	5,748	5,906	4,291	0.73
s9234	25,935	-	21,612	11,498	8,568	8,262	6,672	0.81
s13207	163,100	25,334	32,648	8,517	13,114	12,229	8,683	0.71
s15850	57,424	22,784	26,306	13,873	11,372	11,178	8,904	0.80
s35932	19,393	7,128	-	1,400	7,252	8,005	2,582	0.32
s38417	113,152	89,856	64,976	62,939	30,404	29,897	25,788	0.86
s38584	161,040	38,976	77,372	53,287	28,140	28,526	18,012	0.63
average	-	-	-	-	-	-	-	0.69

## 7. まとめ

本研究では，多重スキャン構造を持つ論理回路を対象としたテストデータ圧縮を行い，LSI テスタに格納するデータ量を少なくすることを目的としている．圧縮後のテストデータ量を少なくする方法として，辞書に格納する単語の種類数を少なくすることにより，辞書のデータ量を少なくするテストパターン変形を提案した．提案手法では，テストデータ中のドントケアビットだけを利用するのではなく，ケアビットに対してもうまく値を変えてやることにより単語の種類数を少なくする．また，テストパターンを増やすことを許容することによりケアビットの制約を緩くし，単語の種類数削減をしやすくする．

提案手法を用いて，ISCAS '89 ベンチマーク回路のテストデータを対象にテストデータ圧縮の実験を行った．実験の結果，SDI 法よりも大幅に単語の種類数を少なくすることができた．圧縮後のテストデータ量についてもすべての回路において SDI 法よりもデータ量を小さくでき，他の従来手法においてもほぼすべての回路においてよい結果を得られた．本研究では，テストパターン変形においてテストパターンを増加しているので圧縮されたテストデータが増加する．しかし，大幅に単語の種類数を削減ができ，結果として，圧縮後のテストデータ量を少なくすることができた．

以上より，提案法の有効性を確認することができた．

## 謝辞

本論文は，著者が三重大学大学院工学研究科博士前期課程に在籍中に行った研究をまとめたものである．本研究を進めるにあたり，懇切丁寧な御指導と御督励を賜った三重大学林照峯教授，北英彦助教授，高瀬治彦助手に感謝いたします．また，日頃熱心に討論していただいた計算機工学研究室の皆様方にお礼申し上げます．

そして，貴重な時間を割いて本研究論文の査読をして頂いた情報処理研究室の篠木剛助教授に感謝いたします．

最後に，本論文をまとめるにあたり，助言，討論，その他お世話になった全ての方々に感謝いたします．

## 参考文献

- [1] 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. Computer-Aided Design of Integrated Circuits and Systems, Vol.14, No.12, pp.1496-1504, 1995.
- [2] I. Hamzaoglu and J. H.Patel, Test Set Compaction Algorithms for Combinational Circuits, Proc. Int. Conf. On Computer-Aided Design, pp.283-289, 1998.
- [3] A. Jas, J. Ghosh-Dastidar and N. A. Touba, Scan Vector Compression /Decompression Using Statistical Coding, Proc. 17th VLSI Test Symposium, pp.114-120, 1999.
- [4] I. Bayraktaroglu and A. Orailoglu, Test Volume and Application Time Reduction through Scan Chain Concealment, Proc. 38th Design Automation Conference, pp.151-155, 2001.
- [5] A. Chandra and K. Chakrabarty, Frequency-Directed Run-Length (FDR) Codes with Application to System-on-a-Chip Test Data Compression, Proc. 19th VLSI Test Symposium, pp.42-47, 2001.
- [6] E. H. Volkerink, A. Khoche and S. Mitra, Packet-based Input Test Data Compression Techniques, Proc. Int. Test Conference, pp.154-163, 2002.
- [7] S. M. Reddy, K. Miyase, S. Kajihara and I. Pomeranz, On Test Data Volume Reduction for Multiple Scan Chain Designs, Proc. 20th IEEE VLSI Test Symposium, pp.103-108, 2002.
- [8] L. Li and K. Chakrabarty, Test Data Compression Using Dictionaries with Fixed-Length Indices, Proc. 21th IEEE VLSI Test Symposium, pp.219 -224, 2003.

- [9] T. Hayashi, H. Yoshioka, T. Shinogi, H. Kita and H. Takase, Test Data Compression technique Using Selective Don't-Care Identification, Proc. of Asia and South Pacific Design Automation Conference 2004 (ASP-DAC 2004), pp.230-233, 2004.
- [10] 加賀博史, テスト・クライシスの処方箋, Design Wave Magazine, 2004 年 2 月号, pp.40-48, 2004.
- [11] 稲垣康善, 論理回路とオートマトン, 株式会社オーム社, 2000.
- [12] 藤原秀雄, コンピュータの設計とテスト, 工学図書株式会社, pp.137-147, 1990.

## 発表論文

- (1) 山下 智之, 高瀬治彦, 北英彦, 林照峯, “LSI テストデータの段階的な圧縮手法に関する一考察”, 平成 17 年度電気学会東海支部連合大会講演論文集, O-297, 2005.
- (2) 山下 智之, 高瀬治彦, 北英彦, 林照峯, “テストパターン変形を用いた LSI テストデータ圧縮の一手法”, 平成 18 年度電気学会東海支部連合大会講演論文集, O-354, 2006.
- (3) 山下 智之, 高瀬治彦, 北英彦, 林照峯, “テストパターン変形に基づく LSI テストデータ圧縮の一手法”, 平成 18 年度三重地区計測制御研究講演会講演論文集, P-17, 2006.