設計自動化49－10
(1989.10.25)

# 状態遷移記述を利用した順序回路テスト生成手法

## Test Pattern Generation for Sequential Circuits using State Transition Descriptions

中田　恒夫
(富士通研究所)

TSUNEO NAKATA
FUJITSU LABORATORIES LTD.

あらまし　状態遷移記述を利用した同期式順序回路のテスト生成手法について報告する。回路合成によって自動的に作られる回路は、規模は小さいがスキャン設計の適用がなされない場合が多いため自動テスト生成が難しくなる。他方で、合成が自動化され人間が論理を追跡しにくいことから、自動テスト生成の段階で高い故障検出率が要求される。本論文では、ある状態において検出可能な故障、遷移先を変更する故障、全く影響を与えない故障の集合を求め、仕様記述に与えられた状態遷移に従って、検出可能な故障の集合を大きくしていくテスト生成手法について述べている。更に、本テスト生成手法をより有効に活用する、付加回路量の小さいテスト容易化手法についても考察を加える。

Abstract　This paper presents an automatic test pattern generation method for synchronous sequential circuits using state transition descriptions. Logic synthesis system generates relatively small circuits so far, whose automatic test pattern generation, however, tends to be very difficult since scan designs cannot be economically applied. Our method is based on the partitioning of fault sets at the states. Along with the state transitions specified by designers, the detectable fault set is expanded until the desirable fault coverage is attained. The designs for testability are also presented which are tightly coupled with the test pattern generation method.

## 1．はじめに

ＡＳＩＣ技術の進歩に伴い、論理合成に対する要請が高まってきている。

組合せ回路を対象とした論理合成技術は近年大きな進歩をとげており、数百ゲート程度の回路では、すでに人手設計をしのぐ品質の回路を自動的に作り出すことが可能である。

順序回路については、状態遷移記述からの合成の研究が今盛んに行なわれている。現在主流を占めているのは、状態割り当てによって状態遷移記述を二段論理回路とレジスタに分け、組合せ回路の合成技術を用いて二段論理回路を簡単化する手法である。

近い将来、制御回路のように再利用がしにくい回路に対しては、論理合成の適用が広く行なわれることが予想される。

一方、テスト生成に目をむけると、組合せ回路を対象とした手法に関しては実用レベルに達しているのに対し、順序回路を扱う手法に関しては未だに研究段階にある。このため、大規模順序回路においてはスキャン設計を採用し、テスト容易化を施すことが一般的である。

ところが、状態遷移記述から自動的に合成された同期式順序回路は比較的小規模であるために、スキャン化することが経済的に許されない場合が多くなる。このため、スキャンを前提としない順序回路用テスト生成手法が不可欠になる。

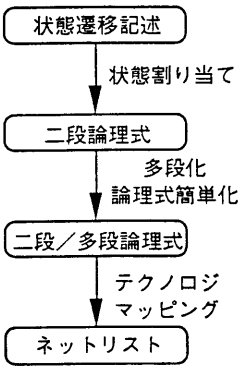


図1
状態遷移記述からの
回路合成手順

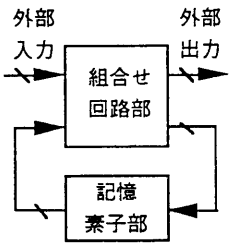


図2
Mealy 型の
同期式順序回路

本稿では、状態遷移記述から回路合成で作られた回路を対象とするテスト生成手法について述べる。本手法は、状態遷移に関する情報を利用して、時間方向のバックトラックを行なわずにテストパターンを生成することを特徴としている。まず、第二章で状態遷移記述からの論理合成の手順を述べ、第三章で合成された回路のテスト生成で問題となる点を指摘する。次に、第四章で本手法について詳細な説明を行なった後、第五章では状態遷移記述レベルでのテスト容易化法について、回路量の評価を含めて考察する。

## 2．順序回路の合成

前章で述べたように、順序回路の合成は研究段階にあり様々なアプローチが採られている。ここでは、制御回路の動作を表現した状態遷移記述からの回路合成を図１の手順で行なう場合を考える。また、合成される回路はMealy型の同期式順序回路（図２）であるとする。

まず最初に、状態遷移記述の中でシンボルで表現されている「状態」に対して二進値パターンを割り当てる「状態割り当て」処理を行なう。Mealy型の順序回路では、割り当てられた二進値パターンが合成される回路のフリップフロップの値そのものとなる。また、状態遷移記述の中に割り当てられたパターンを埋め込んだものは、組合せ回路部をカバー表現による二段論理式で表現したものになる。

一般に、状態割り当てで生成された二段論理式には冗長性が含まれている。そこで、回路量を小さくするために組合せ論理の簡単化処理を施す。組合せ回路部をＰＬＡで構成する場合には二段論理式の簡単化処理［１］、通常のゲート回路で構成する場合は更に多段化、多段論理式の簡単化［２、３、４］を実行する必要がある。

最後に、簡単化によって冗長性が除かれたコンパクトな論理式から、合成される回路のテクノロジに合わせてゲート回路への対応付けを行なう「テクノロジマッピング」を行なう。テクノロジマッピングの出力形式は一般には多段の論理式ではあるが、各カバーが実際のライブラリに現われるマクロ素子に直接対応しており、ネットリストとほぼ等価な形となっている。

図３のような状態遷移図で表される動作から回路が合成される様子を図４に示す。

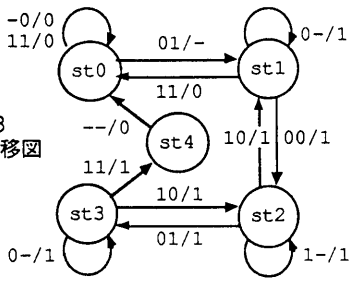


図３
状態遷移図

## 3．回路合成とテスト生成

### 3.1. 回路合成とテスト

状態遷移記述からの回路合成が広く用いられるようになると、テストに関して従来とは異なる新たな問題を生じる。本章では、この問題を指摘し、回路合成システムと結びついたテスト生成における要件を列挙する。

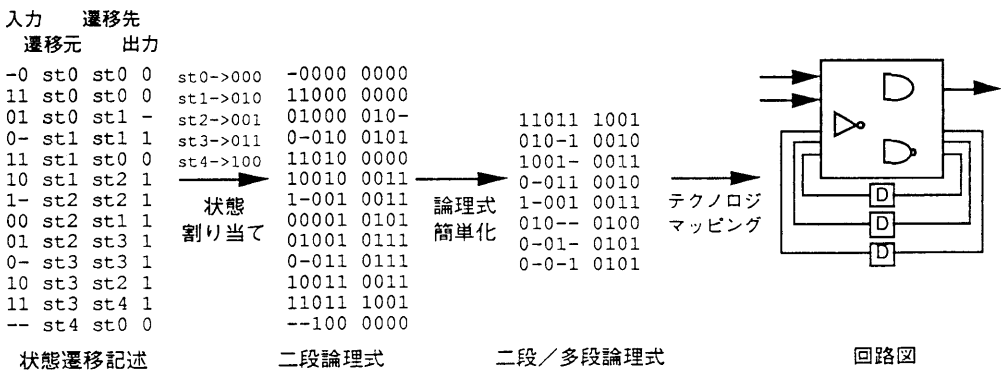


図４　順序回路自動合成の例

### 3.2. 処理の完全自動化

従来の回路では、高い故障検出率を実現するためには人手の介入が必要となることが多かった。すなわち、ランダムパターンやアルゴリズムを用いても故障検出率が目標値に達しなかったときには、設計者やテスト技術者が回路を解析して、パターンの作成や冗長性の検出を行なっていた。

ところが、回路が自動的に作られるようになると、状態割り当てや論理式簡単化の段階で、元の記述が持っていた「回路の特徴」が失われてしまい、回路の解析が困難になる場合が多くなる。

このため、回路合成システムとテストが連携する場合は、テスト生成の完全自動化が絶対的な条件となる。

### 3.3. 非スキャン回路への対応

状態遷移記述から合成される回路は、一般に制御系回路でありそれほど大きな規模にはならない。現在の合成システムの能力の制約もあり、扱うべき回路の規模は、組合せ回路部が千ゲート以内、状態レジスタの個数が１０個以内に抑えられる。

現在、テスト生成を完全に自動化するには、スキャン設計の導入が不可欠であるが、このような規模の回路に対してスキャンを適用することは主に回路量の点で問題である。従って、テスト生成の側で非スキャン回路への対応を図らなくてはならない。

### 3.4. テスト容易化への対応

全スキャンがコストの面で採用できない一方で、回路合成の自動化により、テスト容易化設計の導入はむしろ容易になっている。

従来は、設計とテスト容易化が独立して存在し、設計者が論理設計を終えた回路に対してテスト容易性を測定して必要に応じてテスト容易化手法を適用することが一般的であった。

しかし、回路合成システムを用いると、初期の段階から設計対象が設計者の手を離れ、機械にすべてが委ねられるようになることから、設計の様々な段階でテスト容易化のための施策を行なうことが可能となる。

状態遷移記述から回路合成を行なう場合は、状態遷移記述自体、状態割り当て後に得られる二段論理式（ＰＬＡ）、多段論理式、ゲート回路の各段階におけるテスト容易化が考えられる。

現実には、各設計レベルにおいて様々なテスト容易化手法、ならびにテスト容易化による影響ー回路量、回路遅延量、故障検出率に対する見積りの方法を準備しておき、設計者との会話によってその設計に最も適した容易化手法を適用することになろう。

以上から、本稿では次に並べた事柄を考慮にいれたテスト生成手法を詳説する。

（１）状態遷移記述に含まれる情報の利用
（２）回路構造を利用した処理の高速化
（３）非スキャンのテスト容易化手法との連携

## ４．状態遷移記述を利用したテスト生成

### 4.1.回路条件

本稿で提示するテスト生成手法は、状態遷移記述から合成される回路を対象としており、対象回路に次のような限定条件をつけている。

（１）故障モデルが単一縮退故障であること
（２）組合せ回路部と記憶素子部が完全に
　　分離されたMealy 型回路であること
（３）リセット状態を持つこと

（３）の条件から、テスト生成を開始する段階で状態を表すフリップフロップの値はリセット状態として定義された値にセットされているものと考える。また、対象故障は（１）で示したように単一縮退故障であり、組合せ回路部の信号線、ならびに外部入出力ピンに挿入する。すなわち、フリップフロップ内部の故障は扱わない。さらに、（２）から記憶素子部の外側にはゲート間フィードバックによる値の保持がないことが仮定される。

### 4.2. 基本原理

従来の、順序回路用テスト生成アルゴリズムの多くは対象故障を定めておいて、それを検出するパターンを求めようとするものである［５、６］。

これに対し本手法では、ランダムパターンによるテスト生成と同様に、故障の集合に対して処理を施していく。

まず、状態遷移に応じて検出可能性、状態遷移への影響の２点に従って故障の集合を分割する。その状態遷移において検出できない故障で状態遷移を変化させないものについては、次の状態遷移において検出可能性を調べる。一方、状態遷移を変えてしまう故障については処理を後に回し、通常の状態遷移の流れの中では検出できないことを確認した上で、故障がある場合の遷移とない場合の遷移を比較する。

このように、本手法では時間の流れに沿ってテストパターンを作っていくことから、時間方向を遡る処理、時間方向のバックトラックが起こらないという特徴を持つ。

### 4.3. 故障集合の分割

まず、ある状態、ある入力パターンの下で信号線$x$の故障の影響が外部出力、フィードバック出力に現われるかどうかを表す故障検出関数$O_x$、$S_x$を考える。$O_x$、$S_x$は次の式で定義される。なお、$dF/dx$は論理関数$F$の$x$に関するブール微分を示す。

$$O_x = dO_1/dx + dO_2/dx + \cdots + dO_n/dx$$
$$S_x = dS_1/dx + dS_2/dx + \cdots + dS_m/dx$$

$O_i$：外部出力の論理関数

$S_j$：フィードバック出力の論理関数

一つの状態遷移において各故障を次の三つの集合に分類する（図5）。なお、信号線$x$の論理値の否定を$\nu$とする。

（1）検出可能な故障：

$$F_o = \{ SA\nu \text{ on } x \mid O_x = 1 \}$$

（2）故障により遷移先状態が変化する故障：

$$F_s = \{ SA\nu \text{ on } x \mid O_x = 0 \wedge S_x = 1 \}$$

（3）全く故障の影響が現われない故障：

$$F_u = \{ SA\nu \text{ on } x \mid O_x = 0 \wedge S_x = 0 \} \cap \{ SA(\bar{\nu}) \text{ on } x \}$$

状態遷移において入力にドントケアがない場合には故障シミュレーションによってこれらの集合が求められる。ドントケアがある場合には、組合せ回路部に対して組合せ回路用のテスト生成アルゴリズムを適用するか、あるいはランダムパターンなどをドントケア入力に加えて故障シミュレーションを行なうことで求めることができる。

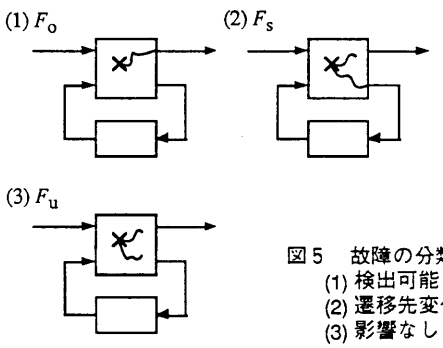


図5　故障の分類
(1) 検出可能
(2) 遷移先変化
(3) 影響なし

### 4.4. 分割された故障集合に対する処理

本手法では、状態遷移に沿って故障集合の分割を進め、検出可能な故障を増やしていく。リセット状態においては分割の対象となる故障の集合は全故障の集合に等しいが、状態遷移を進めるにつれて扱うべき故障の数を減らすことができる。遷移先の状態における故障集合は遷移元で分割された集合に対して処理を施すことによって求められる。以下に、回路内のある故障$f$が属している集合に応じて行なうべき処理を示す。

（1）$f \in F_o$

$f$は検出可能であるため、この故障を故障集合から除去する。

（2）$f \in F_s$

$f$により状態遷移自体が変化する。このような故障を検出するには故障のある場合とない場合それぞれについて入力系列に対する状態遷移を追うことが不可欠であり、一般に検出が困難である。ここでは、一旦$f$を処理すべき故障のリストから除いておく。そして、リセット状態からの、他の正常な状態遷移の中で検出可能であるかどうかを調べ、あらゆる場合に検出不能であることが判明した場合に限り、状態遷移の追跡を行なう。

（3）$f \in F_u$

$f$の影響が現われないため、次の状態は状態遷移記述に記載された状態そのものである。$f$が未検出で残されることから、$f$を無条件で次の状態で処理すべき信号線の集合に加える。

図6は、これらの処理が状態遷移の中でどのように行なわれるかを示している。

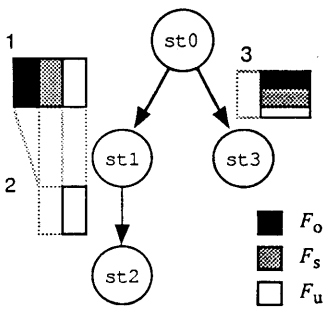


図6　状態遷移に伴う故障集合の変化

まず、状態st0からst1への遷移において、扱うべき故障集合が三つに分割される。このうち$F_0$は無条件で故障集合から除かれる（図中１）。また$F_s$に関しては、st1への遷移自体が起こらないことから、次の状態で扱うべき故障集合から除去される。st1からst2への遷移でも同様の処理を行なう(図中２)。このように、まずある状態からの遷移に着目して、depth-firstに状態遷移を繰り返しながら故障を検出していく。

次に、st1側の遷移に対する処理がすべて終わった段階で、st0からの別の遷移に着目する。この際に、st1側の遷移で未検出故障の集合と、状態遷移を変更するために途中で除去された故障の集合をマージして新たな故障集合を構成する（図中３）。

このように、リセット状態から状態遷移記述に従ってすべての到達可能な状態への遷移を行ない、検出可能な故障を求めていく。

すべての遷移を終えた段階で、$F_s$、$F_u$に残されている故障については従来の順序回路用テスト生成手法を適用してテストパターンを求める。

通常は、組合せ回路用テスト生成アルゴリズムを用いて、故障を検出できる状態と入力パターンをまず求め、故障がないときに初期状態から検出可能な状態への遷移を実現する。故障が未検出であったことから途中で遷移先が変化しているはずであるが、多くの場合、遷移先が変化していれば検出可能な状態への遷移の途中で出力値に差を生じ、故障が検出できる［６］。

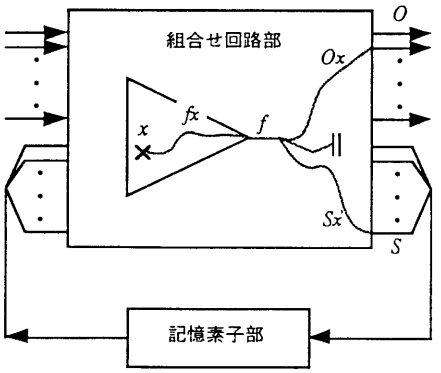


図７　分岐点に着目した高速化の概念図

4.5. 高速化のための方策

故障集合の分割において故障シミュレーションを用いる場合は、回路構造、特に分岐構造に着目した代表故障の絞り込み［７］が有効に機能する。

任意の組合せ回路は、分岐点で回路を切ることによって樹枝状の部分回路に分解できる。このとき、樹枝状回路の根に相当する端子、すなわち外部出力端子と分岐点の故障が検出可能であることが、その部分回路の故障が検出可能であることの必要条件となっている。そこで、テスト生成の対象となる代表点として外部出力端子と分岐点のみを考え、処理量の削減を図る。

まず、代表点の故障の集合に対して分割を行なう。ある代表点の故障が検出可能である場合は、樹枝状回路の各故障に対して検出可能であるかどうかのチェックを行なう。

いま、樹枝状部分回路の先端の信号線を$f$、回路内部の任意の信号線を$x$で表す。すでに定義した故障検出関数の記法を用いて、信号線$x$に存在する故障の影響が$f$に現われることを示す関数を$f_x$と書く。部分回路が樹枝状であることから、次の式が成立する。

$$O_x = f_x \cdot O_f \ , \ S_x = f_x \cdot S_f$$

ここで、$f_x$の値は部分回路の回路量に比例した時間で計算可能である。

上の式を用いると、代表点の故障検出関数の値に応じて図７に示す処理を行なうことで樹枝状部分回路の故障検出性をチェックできる。$O_f$、$S_f$の値が０であるときには、部分回路内部の故障について考える必要がなくなるため、処理の効率化がなされる。

4.6. 例題

図８の回路に対して、実際にテスト生成を適用してみる。まず、テスト生成器に与えられるのは図８の回路構造と図９の状態遷移記述である。ここでは、故障集合すべてを考えるのではなく、本手法の特徴が現われる故障として信号線$g$の故障を対象とする。

まず、初期状態st0からの遷移のうち、状態遷移記述の最初に現われる遷移に着目する。この遷移では外部入力にドントケアがないため、組合せ回路部の入力がすべて確定している。ここで通常の故障シミュレーションを実行することによって、回路内のすべての故障を三つに分類できる。信号線$g$の0,1縮退故障はそれぞれ$F_u$、$F_s$に属する。

次に、遷移先状態からの状態遷移を記述から取り出してきて同様の処理を続ける。このとき、扱うべき故障集合は最初の遷移における$F_u$となり、信号線$g$の1縮退故障が故障シミュレーションの対象から除かれる。ただし、本例では遷移先状態がst0そのものであるので処理を進める必要がない。そこで次に扱うべき処理として、初期状態st0からの別の遷移として二番目の状態遷移を採用する。

st0からst1への状態遷移でも組合せ回路部のすべての入力が確定しており、故障シミュレーションによって故障が分類される。このとき、対象とすべき故障集合は先の遷移における$F_u$と$F_s$の和集合で与えられる。従って、信号線$g$の1縮退故障は再び故障シミュレーションの対象となる。

このように、比較的故障検出が難しい、状態遷移を変更させる故障は後回しにされ、他の「正規の」状態遷移の中で検出可能であるかどうかをチェックする。信号線$g$の1縮退故障は、st0からst3への状態遷移において検出可能である。

一方、状態遷移を変更させる故障の中で、決して外部出力に直接影響が現われないものは、すべての状態遷移に対する処理を終了した段階で、なお集合$F_s$に属している。このような故障に対しては、従来のテスト生成手法を適用する必要がある。ただし、多くの場合は故障の影響範囲がごくフィードバック出力に近いところに限られるため、変化する状態レジスタからの入力の縮退故障の検出問題を解決すればよい。例えば、信号線$y$の故障のテスト生成問題は、外部入力$c$の問題と等価である。

## 5．状態遷移記述からのテスト容易化

### 5.1. 未定義状態

前章で述べたアルゴリズムにおいて、テスト生成の効率を低下させる大きな要因として、状態遷移記述に現われてこない状態「未定義状態」の存在が挙げられる。

一般に状態数が$n$個あるとすると、状態を表すために必要なフリップフロップの数$N$は、$\lceil \log_2 n \rceil$となる。ところが、$N$個のフリップフロップで表現できる状態の数は、$2^N(\geq n)$であるため、実際には状態遷移記述に現われてこない状態が存在する。例えば、図3では状態数が5であるから、フリップフロップは最低3個必要である。これに図4の状態割り当てを行なうと、101、110、111で表される3個の状態が未定義状態となる。

未定義状態に関する状態遷移は状態遷移記述で定義されないため、未定義状態に遷移する状態、未定義状態から遷移する状態は回路合成結果に依存し予測ができない。従って、未定義状態にあるときだけ検出可能な故障や正常な状態から未定義状態に遷移させる故障に対するテストパターンを求めようとすると、時間方向、空間方向のバックトラックを頻繁に生じることになり、生成時間の増大につながる。

### 5.2. テスト容易化設計による解決

未定義状態に関係する故障のような検出が難しい故障に対してテストパターンを高速に求める手法として、スキャン設計に代表されるテスト容易化設計が挙げられる。スキャン設計を用いると、問題が組合せ回路のテスト生成に帰着され根本的な解決とな

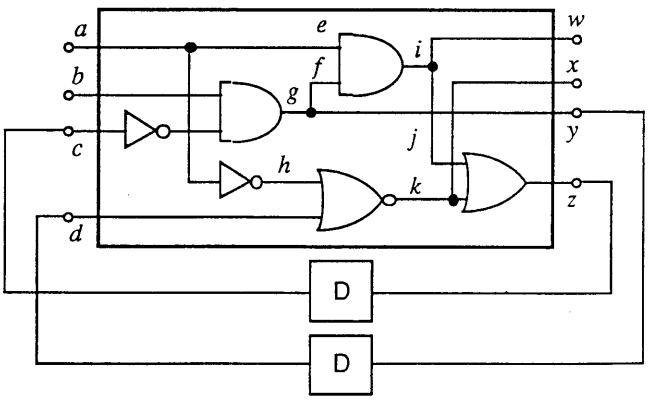


図8　例題回路

```
ab          wx
00 st0  st0 00
01 st0  st1 00
10 st0  st2 01
11 st0  st3 11
-- st1  st0 -0
0- st2  st0 -0
1- st2  st2 01
-- st3  st0 00

        cd,yz
  st0 -> 00
  st1 -> 01
  st2 -> 10
  st3 -> 11
```

図9　例題回路の状態遷移記述と状態割り当て

る。ところが、スキャン設計は付加回路量、付加回路による遅延量が大きく、制御系回路ではコストパフォーマンスの面で採用しにくい。

状態遷移記述から合成された回路であるという特性を活用し、スキャン設計によらないテスト容易化の実現を考える。ここでは、状態遷移記述の段階で、外部入出力端子、状態遷移を新たに付加することによって特に未定義状態に関係した故障の検出性を高める方法について述べる。

### 5.3. 未定義状態検出出力の付加

故障の影響で正常な状態から未定義状態へ遷移する場合、未定義状態であることを判別できればその故障は検出可能である。そこで、正常な状態においては0、未定義状態では1となるような外部出力端子を付加することを考える。

論理の実現法としては二通り考えられる。第一は、未定義状態のみを考慮して論理式を立てて回路に落とす方法、第二は、状態遷移記述の段階で未定義状態を仮定義し、検出出力との関連を記述する方法である。

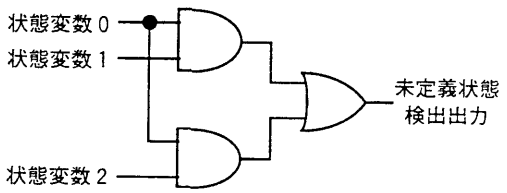


図10　未定義状態検出回路

図4の例題に対して、第一、第二の方法を適用してみる。例題では状態数が5であるため、3個のフリップフロップが必要で、逆に3個のフリップフロップで8個の状態を表現できることから、3個の未定義状態st5、st6、st7が存在することになる。これらに、それぞれ101、110、111のコードを割り当てる。

第一の方法では、3入力1出力の論理式で、入力が上記の3パターンのときに限り1となるようなものを考えて回路化すればよい。一例を図10に示す。一方、第二の方法では、元の状態遷移記述に対して、未定義状態からの遷移を付加する。このとき入力、遷移先、通常の出力値は任意であるとして、未定義状態検出出力のみが1となるようにする。また、通常の状態遷移に対しても、未定義状態検出出力として0を加える。図4の状態遷移記述に対してテスト容易化を施したものを図11に示す。このように書き換えた状態遷移記述に対して合成を行なうことによってテスト容易化が実現される。

一般に、第二の方法の方が回路がコンパクトになる傾向があるが、正規の論理と一緒に回路合成されるため、対象故障が未定義状態検出出力の値を変えない場合があり、故障検出率では第一の方法が優れていると考えられる。

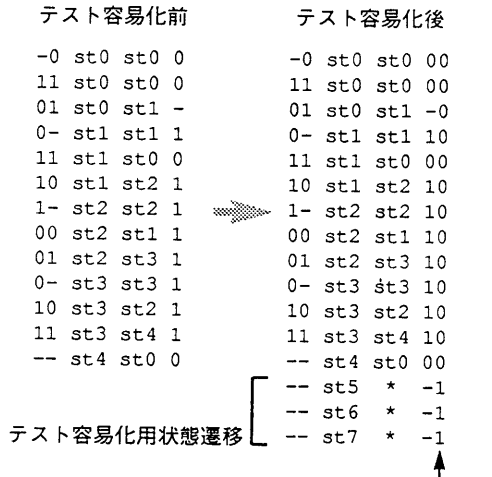


図11
未定義状態検出出力付加による
テスト容易化

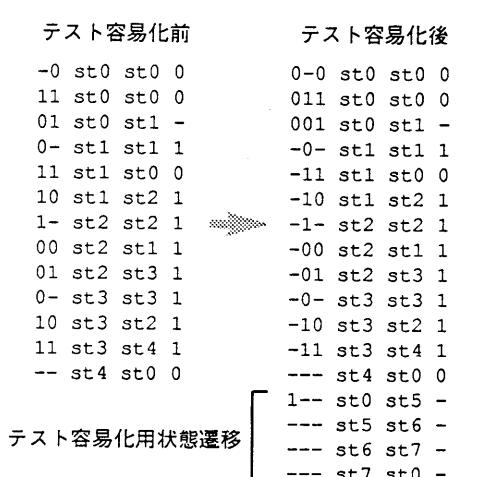


図12
未定義状態遷移入力付加による
テスト容易化

### 5.4. 未定義状態へ遷移させる入力の付加

未定義状態の下でのみ検出可能な故障に対処するためには、回路の動作モードを通常モード、テストモードの二種類に分け、テストモードでは正規の状態から未定義状態へ強制的に遷移させるようにする方法が有効であると考えられる。

回路に対しては、モード切り替えのための外部入力を１本加える。その値に応じて、初期状態からの遷移先が正常な状態であるか、あるいは未定義状態であるかが切り替わるようにする。また、すべての未定義状態を鎖状につなぐ遷移を行なわせることによって、任意の未定義状態へ遷移できるようにする。また、前節で述べた手法を併用して、未定義状態にいる間は未定義状態検出出力が１となるようにする方法も有効であると考えられる。

図４の状態遷移記述に対して、上記の処理を施した結果を図１２に示す。

### 5.5 テスト容易化の影響

MCNCのFSMベンチマーク回路４種に対して、本テスト容易化手法を適用した結果を図１３に示す。ここでは、ベンチマーク回路に対して、状態割り当て、二段論理式簡単化、多段化、多段論理簡単化を順次適用し、結果として得られた多段論理式のリテラル数を比較している。

図１３によると、容易化手法Ａ：未定義状態検出出力の付加、容易化手法Ｂ：未定義状態への遷移入力の付加のそれぞれにおけるリテラル数の増大は、それぞれ１０%、２５%程度となっている。また、論理段数に関してはテスト容易化による変化は認められなかった。

## ６．おわりに

状態遷移記述からの回路合成を前提とした同期式順序回路用テスト生成手法、ならびにテスト容易化手法について述べた。

テスト生成については、状態遷移において、初期状態から状態遷移をたどりながら検出可能な故障を拾い上げていくという方針の下で、故障シミュレーションを基本ツールとし、対象故障の絞り込みによって処理量、記憶量を抑えた手法を提案した。

また、状態遷移記述に記載されていない「未定義状態」の存在が故障検出率の低下、処理量の増大を引き起こす可能性があることを指摘し、これを克服するテスト容易化手法を示した。

現在、テスト生成手法、テスト容易化手法の両者について、基礎的な検討を終え、回路合成実験システムと結合作業を進めている。今後は、両手法について、実回路を対象に評価を行なっていく予定である。

＜参考文献＞


[1] Brayton, et. al.: Logic Minimization Algorithm for VLSI Synthesis, Kluwer Academic Publishers, 1984.

[2] Saldanha, et. al.: Multi-Level Logic Simplification using Don't Cares and Filters, Proc. of 26th DAC, pp. 277-282, 1989.

[3] Bostick, et. al.: The Boulder Optimal Logic Design System, Proc. of ICCAD-87, pp.62-65, 1987.

[4] 松永、藤田：２分決定グラフを用いた多段論理回路の最適化手法、信学技法、Vol. 89, No. 92, CAS-89-10, pp.19-26, 1989.

[5] Marlett: An Effective Test Generation System for Sequential Circuits, Proc. 23rd DAC, pp.250-256, 1988.

[6] Tony Ma, et. al.: Test Generation for Sequential Circuits, IEEE Trans. on CAD, Vol. 7, No. 10, pp. 1081-1093, 1988.

[7] Antreich and Schultz: Accelerated Fault Simulation and Fault Grading in Combinational Circuits, IEEE Trans. on CAD, Vol.CAD-6, No.5, pp.704-712, 1987.


| | | 入力 | 出力 | 状態 | リテラル | 比 |
|---|---|---|---|---|---|---|
| planet | 元回路 | 7 | 19 | 48 | 437 | 1.00 |
| | 容易化Ａ | 7 | 20 | 64 | 478 | 1.09 |
| | 容易化Ｂ | 8 | 20 | 64 | 547 | 1.25 |
| dk16 | 元回路 | 2 | 3 | 27 | 247 | 1.00 |
| | 容易化Ａ | 2 | 4 | 32 | 285 | 1.15 |
| | 容易化Ｂ | 3 | 4 | 32 | 298 | 1.21 |
| ex1 | 元回路 | 9 | 19 | 20 | 242 | 1.00 |
| | 容易化Ａ | 9 | 20 | 32 | 284 | 1.17 |
| | 容易化Ｂ | 10 | 20 | 32 | 344 | 1.42 |
| keyb | 元回路 | 7 | 2 | 19 | 182 | 1.00 |
| | 容易化Ａ | 7 | 3 | 32 | 180 | 0.99 |
| | 容易化Ｂ | 8 | 3 | 32 | 230 | 1.26 |

図１３
テスト容易化に伴う回路量の変化
容易化　Ａ：未定義状態検出、Ｂ：未定義状態遷移