社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2003－ＳＬＤＭ－112　（４）
2003／11／28

状態集合分割を用いる論理シミュレーションによる順序回路のテスト生成

佐野　広和†　　四柳　浩之†　　橋爪　正樹†　　為貞　建臣†

† 徳島大学工学部　〒 770–8506 徳島県徳島市南常三島町 2–1
E-mail: †{zuka,yanagi4,tume,tamesada}@ee.tokushima-u.ac.jp

あらまし　本論文では，状態集合分割を用いた論理シミュレーションによるテスト生成について述べる．本研究で行うテスト生成では，複数の候補ベクトルを作成し，未到達状態へ遷移可能なベクトルをテストベクトルとして採用する．故障検出に有効な状態遷移を得るために，フリップフロップを複数の集合に分割し，重み付けによる状態変化の優先度の設定を行う．状態集合の分割法として，フリップフロップの論理値の制御容易性に基づく分割法と，未検出故障への影響の大きいフリップフロップを回路構造から求め集合分割を行う手法の２つを用いる．本研究では各未検出故障の励起，伝搬に必要なフリップフロップを考慮した集合分割を用いることで，未検出故障の検出に必要な状態遷移を優先する．提案する集合分割法をテスト生成に適用した結果により，集合分割法の有効性について示す．
キーワード　状態集合分割，論理シミュレーション，テスト生成，順序回路

Test Generation for Sequential Circuits by Logic Simulation using State Partitioning

Hirokazu SANO†, Hiroyuki YOTSUYANAGI†, Masaki HASHIZUME†, and Takeomi TAMESADA†

† Faculty of Engineering, Univ. of Tokushima　Minami-Josanjima 2–1, Tokushima, 770–8506 Japan
E-mail: †{zuka,yanagi4,tume,tamesada}@ee.tokushima-u.ac.jp

Abstract　This paper presents a test generation method for sequential circuits by logic simulation using state partitioning. The test pattern generation method utilized in this work selects the test vector which can bring the state of the circuit into a state that can not be reached by the previous vectors. Two state partition methods are proposed: One is the partition method based on the controllability of flip-flops. The other is the partition method based on the relation between undetected faults and flip-flops. To obtain test vectors that bring the circuit into the states required for detecting undetected faults, the new method partitions flip-flops in a circuit by considering whether the flip-flop is in the subcircuits required for fault excitation or in the subcircuits required for fault propagation. The experimental results show the effectiveness of the state partitioning methods.
Key words　state partitioning, logic simulation, test generate, sequential circuit

1. まえがき

近年，半導体技術の進歩により微細加工が可能となり，論理回路の小型化が進み，IC 内に大規模回路を実現できるようになった．そのため回路に故障が発生した際，その故障検出は非常に困難であり，そのテスト生成はますます時間と経費を要する問題となっている．一般に，論理 LSI は順序回路であり，そのテスト生成は組合せ回路に比べてさらに困難である．そのため，順序回路の効率の良いテスト生成法を開発することは重要である．

順序回路の縮退故障を対象とするテスト生成手法の一つにランダムベクトルを用いたシミュレーションによるテスト生成手法がある．文献 [1] では，ランダムベクトルを用いた並列故障シミュレーションによるテスト生成手法が提案されている．また，故障の中には，ある特定の状態に設定しなければ検出できない故障が存在するため，可能なかぎり多くの状態に遷移させることを目的とし，テストパターンを生成する手法が提案されている [2], [3], [6].

これらの論理シミュレーションに基づくテスト生成では，正常回路を対象に広範囲にわたる状態遷移が起こるようなテスト

パターンを生成するが，多くの状態遷移が可能なテストパターンであっても，フリップフロップの値の変化に着目した場合，値が変化しやすいデータパスに属するフリップフロップの値のみ変化し，コントロールパスに属するフリップフロップの値はほとんど変化しないなど，フリップフロップにおける値の変化に偏りが生じる場合があることが知られている [2]．特定のフリップフロップに変化が偏ることを防ぐために，値の変化が起きやすいフリップフロップと値の変化が起きにくいフリップフロップを別々の集合に分割し，テストベクトル生成時に変化の起きにくいフリップフロップの値の変化を優先させる手法が提案されている [2]．

本論文では，フリップフロップを複数の集合に分割することを状態集合分割とよぶ．本論文で状態集合分割を，値の制御容易性により状態集合分割を行う手法と，未検出の縮退故障を対象にし，未検出故障とフリップフロップの位置関係に着目して状態分割を行う回路構造を基にする状態集合分割法の 2 つを提案し，論理シミュレーションを用いるテスト生成手法への適用結果を示す．

本論文の構成を以下に示す．第 2 章では，論理シミュレーションによるテスト生成法の概要を述べる．第 3 章では，フリップフロップ制御容易性分割法と，対象故障とフリップフロップとの位置関係に着目する回路構造に基づく分割法について述べる．第 4 章では，第 3 章で述べた状態集合分割法を適用したテスト生成法について述べる．第 5 章では，提案手法に対する実験結果を述べ，第 6 章で本論文のまとめを述べる．

2. 論理シミュレーションによるテスト生成

論理シミュレーションに基づいたテスト生成では，正常回路を対象に広範囲にわたる状態遷移を引き出すことを目的とする [2], [3]．検査困難な故障の検出には，到達困難な状態への遷移が必要な場合があるため，より多くの未到達状態へ遷移させることで，故障を検出可能な場合がある．そのため，未到達状態に遷移させることを目的としたテスト生成手法が提案されている [3]．

本研究で行うランダムパターンを用いるテスト生成では，テストベクトルの選択時に複数の候補ベクトルを作成し，未到達状態へ遷移させることの可能なベクトルをテストベクトルとして採用する．候補ベクトルの生成には乱数ベクトルを初期集合とした遺伝的アルゴリズムを用いる．生成した候補ベクトルを印加した場合の遷移先の状態を求め，それが未到達状態かどうかを判定する．未到達状態へ遷移可能な候補ベクトルが存在すればそのベクトルをテストベクトルとして採用し，存在しなければ直前の状態と異なる状態へ遷移させるベクトルをテストベクトルとして採用する．

図 1 にテスト生成時における入力とそれにより遷移する状態例を示す．

初期状態を S_0 とし，初めに V_1 がテストベクトルとして採用され，状態が S_1 に遷移したと仮定する．このとき，3 つの候補ベクトル生成により，図 1 の V_{2a}，V_{2b}，V_{2c} が生成された場合，各候補ベクトルによる遷移先の状態から，未到達状態 S_{2a} への遷移を行う候補ベクトル V_{2a} をテストベクトル V_2 として採用する．本研究で用いるテスト生成では，上記のようなテストベクトルの生成をすべての故障が検出されるか，あらかじめ与えられたテスト系列長に達するまで繰り返し行う．

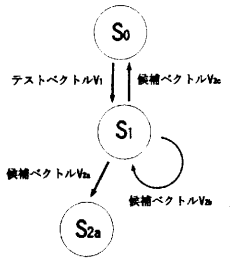

図 1　テスト生成時におけるベクトルと状態遷移の例

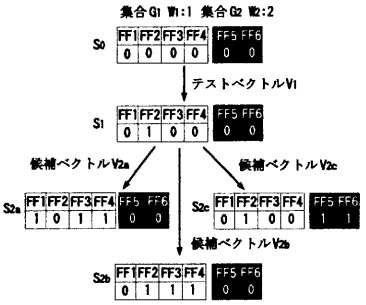

図 2　状態集合分割テスト生成例

3. 状態集合分割法

3.1 状態集合分割を用いるテスト生成の概要

第 2 章で述べた論理シミュレーションによるテスト生成では，可能なかぎり未到達状態へ遷移可能な候補ベクトルをテストベクトルとして採用することで，多くの状態への遷移が可能なテストパターンの生成を行う．しかし，回路内のフリップフロップには値の変化が起きやすいフリップフロップと，値の変化が起きにくいフリップフロップが存在する場合がある．そのため，未到達状態へ遷移させることを目的としたテスト生成を行った場合でも，特定のフリップフロップに変化が集中し，他のフリップフロップの値が変化していない可能性がある．文献 [2] では，値の変化が起きやすいフリップフロップとそうでないフリップフロップとを別々の集合に分割し，テスト生成時に変化の起きにくいフリップフロップの集合における値の変化を優先する状態集合分割法が提案されている．

フリップフロップの値の変化の容易性に応じて状態集合の分割を行い，変化が困難なフリップフロップの値の変化を優先させることで，テスト生成時に特定のフリップフロップに変化が集中することを避けることが可能である．

図 2 に状態集合分割を用いたテスト生成例を示す．

回路内に 6 個のフリップフロップが存在し，$FF1$, $FF2$, $FF3$, $FF4$ は値の変化が起こりやすいフリップフロップ，$FF5$, $FF6$ は値の変化が起こりにくいフリップフロップであると仮定する．状態集合分割により図 2 に示すように，値の変化が起こりや

すいフリップフロップの集合 G_1 と，値の変化が起こりにくいフリップフロップの集合 G_2 の 2 つの集合に分割する．次に，G_1，G_2 それぞれの集合に重み $W_1 = 1$，$W_2 = 2$ を割り当てる．初期状態 S_0 からのテスト生成により，V_1 が採用されたと仮定し，状態 S_0 からテストベクトル V_1 により状態 S_1 に遷移すると仮定する．さらに乱数を用いて 3 つの候補ベクトル V_{2a}，V_{2b}，V_{2c} を作成する．このとき状態 S_1 から，3 つの候補ベクトルによりそれぞれ，状態 S_{2a}，S_{2b}，S_{2c} に遷移したとする．この場合，状態 S_{2a}，S_{2b}，S_{2c} はすべて未到達状態である．しかし 3 状態の内，状態 S_{2a}，S_{2b} は集合 G_1 の値のみの変化であり，集合 G_2 の値は変化していない．状態 S_{2c} は集合 G_2 のみの値の変化で，集合 G_1 の値は変化していない．この場合，集合 G_2 の重みの方が集合 G_1 の重みよりも大きいため，集合 G_2 の値を変化させることの可能な候補ベクトル V_{2c} をテストベクトル V_2 として採用し，状態 S_{2c} をテストベクトル V_2 による遷移先状態 S_2 とする．この手法により，複数の候補ベクトルが未到達状態への遷移を行う場合に，変化の起きにくいフリップフロップの値を変化させることのできる候補ベクトルをテストベクトルとして採用できる．

3.2 フリップフロップ制御容易性による状態集合分割法

フリップフロップ制御容易性分割法では，状態変化の起きにくいフリップフロップの選別を行うために，N 個のランダムベクトルを用いて論理シミュレーションを行う．その際に，フリップフロップ FF_i が 0，1 に設定される回数を，それぞれ N_i^0，N_i^1 とする．これにより，求められるフリップフロップ FF_i の 0 及び 1 の制御確率 $CC0_i$，$CC1_i$ を次に示す．

$$CC0_i = \frac{N_i^0}{N} \tag{1}$$

$$CC1_i = \frac{N_i^1}{N} \tag{2}$$

式 (1)，式 (2) より，フリップフロップ FFi の制御容易性 $FF_{bias}(i)$ は，式 (3) のように表される．

$$FF_{bias}(i) = |CC0_i - CC1_i| \tag{3}$$

FF_{bias} の値が 0 に近いほど，フリップフロップが 0，1 に設定される確率が同程度であるということを表している．また，FF_{bias} の値が 1 に近いほどそのフリップフロップの論理値が 0 または 1 に偏る確率が高く，値が制御しにくいということを表している．FF_{bias} の値が大きいフリップフロップの変化を優先させることで，制御しにくいフリップフロップの状態を変化させることの可能な候補ベクトルを採用することが可能となる．

提案手法では，$FF_{bias}(i)$ の値が 0.2 以下，0.2〜0.4，0.4〜0.6，0.6〜0.8，0.8〜1.0 のフリップフロップ FFi を集合 G_1，G_2，G_3，G_4，G_5 にそれぞれ分割する．また，集合 G_j の重み W_j は，$W_1 > W_2 > W_3 > W_4 > W_5$ となるように割り当てる．以下に，フリップフロップ制御容易性分割の手順を示す．実験においては，$N = 1000, W_1 = 5, W_2 = 4, W_3 = 3, W_4 = 2, W_5 = 1$ という値を用いた．

（1） 1000 個のランダムベクトルを用いて論理シミュレーションを行い，N_i^0，N_i^1 を計測

FF1	FF2	FF3	FF4	FF5	FF6
FFbias(1)	FFbias(2)	FFbias(3)	FFbias(4)	FFbias(5)	FFbias(6)
0.5	0.2	0.6	0.5	0.1	0.9

図 3 フリップフロップ制御容易性分割適用例

G1	G2			G3	
FF6	FF1	FF3	FF4	FF2	FF5
W1:3	W2:2			W3:1	

図 4 フリップフロップ制御容易性分割による分割例

（2） 式 (1)，式 (2) を用いて，$CC0_i$，$CC1_i$ を計算

（3） 式 (3) を用いて，フリップフロップ FFi の制御容易性 $FF_{bias}(i)$ を計算

（4） FF_{bias} の値を分割条件と比較し，各フリップフロップを 5 つの集合に分割

（5） 分割時の FF_{bias} の値が大きい集合から順に，重み W_1〜W_5 を割り当て

図 3，図 4 に，フリップフロップ制御容易性分割による分割例を示す．

回路内の 6 つのフリップフロップが，図 3 のような FF_{bias} の値になったと仮定する．この場合，各フリップフロップを分割条件にしたがって分割を行った結果を図 4 に示す．

各 FF_{bias} の値がどの範囲の集合に当てはまるかによって，分割を行う．集合 G_1 には，$FF_{bias}(6)$ の値が 0.8〜1.0 以下に含まれるため $FF6$ を配分する．同様に集合 G_2 は，それぞれの FF_{bias} の値が 0.4〜0.6 以下に含まれる FF_1，FF_3，FF_4 を配分する．同様に G_3 には，FF_2，FF_5 を配分する．集合 G_j の重み W_j は，集合 G_1 に一番大きい重みを割り当て，$W_1 = 3$ とする．次に集合 G_2 に $W_2 = 2$ を割り当て，G_3 に $W_3 = 1$ を割り当てる．重みの大きい集合の状態変化を優先することにより，論理値の偏りの大きいフリップフロップの値を変化させることが可能なベクトルを採用できる．これにより，フリップフロップの値が偏らない状態遷移を得ることが可能となる．FF_6 は $FF_{bias}(6)$ の値が 0.9 であるため，偏りが大きく変化が起きにくい．そのため，集合 G_1 に $W_1 = 3$ を割り当てることにより FF_6 の値を変化させることの可能なベクトルを優先的に採用する．

3.3 対象故障との位置関係に着目する回路構造に基づく状態集合分割法

フリップフロップ制御容易性分割では，分割した各フリップフロップの集合に異なる重みを与えることで，一部のフリップフロップのみに値の変化が偏るテストパターンを生成することを防いでいる．

一方，縮退故障の中には，故障検出に必要な状態が少ないため，検査困難な故障が存在する．フリップフロップ制御容易性分割を用いたテスト生成のみでは，それらの故障の検出に必要な状態への遷移を行うテストベクトルを生成することが困難であり，生成するテストベクトル数を増加させても検出故障数が増えない可能性がある．文献 [2] では，未検出故障の検出に必

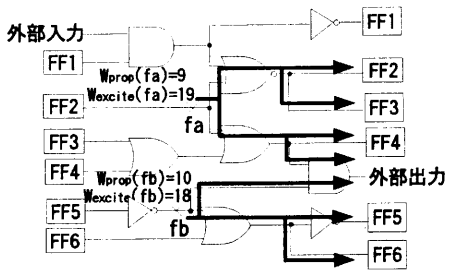

図 5　W_{prop}, W_{excite} の計算例

要な状態遷移を行うテストベクトルの生成のために，故障とフリップフロップの関係を基に状態集合の分割を行っている．

本研究では，検査困難な故障に着目した状態集合分割法として，対象故障に与える影響の大きいフリップフロップの変化を優先させる状態集合分割手法を提案する．ここで対象とする検査困難な故障とは，フリップフロップ制御容易性分割を用いたテスト生成により得られたテストベクトルを用いた故障シミュレーションで新たに故障を検出できない状態が続いた時点の未検出故障である．各故障の検出に必要なフリップフロップとして，故障の励起に必要とされるフリップフロップと故障の伝搬に必要とされるフリップフロップの区別を行い，未検出故障の検出により適した状態遷移を行うテストベクトルの選択を可能とすることを目的とし，各故障 f に，フリップフロップと外部出力との位置関係により決まる値 $W_{prop}(f)$, $W_{excite}(f)$ を設定する．W_{prop} は，故障を外部出力まで伝搬するために必要なフリップフロップの重要度を表し，伝搬経路の少ない故障ほど値を高くする．また W_{excite} は，故障を励起するために必要なフリップフロップの重要度を表し，伝搬経路の多い故障ほど値を高くする．$W_{prop}(f)$, $W_{excite}(f)$ は以下の式で計算する．

$$W_{prop}(f) = N_{out} \times 2 - N_{po}(f) \times 2 - N_{pFF}(f) \qquad (4)$$

$$W_{excite}(f) = N_{out} \times 2 + N_{po}(f) \times 2 + N_{pFF}(f) \qquad (5)$$

ここで，式 (4)，式 (5) の N_{out} は外部出力，疑似外部出力の合計数であり，$N_{po}(f)$ は故障 f の外部出力への伝搬経路数，$N_{pFF}(f)$ は故障 f の疑似外部出力への伝搬経路数を表す．

故障の検出を行うためには，故障の影響を外部出力へ伝搬させる必要がある．そのため，疑似外部出力への伝搬の重要度よりも，外部出力への伝搬の重要度を高くするために，$N_{po}(f)$ の値は 2 倍し，$N_{pFF}(f)$ の値よりもフリップフロップの集合分割時に与える重みへの影響を大きくしている．式 (4)，式 (5) を用いて，故障ごとに W_{prop}, W_{excite} の計算を行う．

図 5 に，W_{prop}, W_{excite} の計算例を示す．

図 5 の fa, fb は回路内の未検出故障を表す．図 5 より，N_{out} は 7 である．このとき，$N_{po}(fa) = 1$, $N_{po}(fb) = 1$, $N_{pFF}(fa) = 3$, $N_{pFF}(fb) = 2$ である．これより，式 (4)，式 (5) を用いて $W_{prop}(fa) = N_{out} \times 2 - N_{po} \times 2 - N_{pFF} = 14 - 2 - 3 = 9$, $W_{excite}(fa) = N_{out} \times 2 + N_{po} \times 2 + N_{pFF} = 14 + 2 + 3 = 19$ となる．同様に，$W_{prop}(fb) = 14 - 2 - 2 = 10$, $W_{excite}(fb) = 14 + 2 + 2 = 18$ となる．$W_{prop}(fa)$ が $W_{prop}(fb)$ に比べ低く，$W_{excite}(fa)$ が $W_{excite}(fb)$ に比べ高い．これは，fa は fb よりも伝搬しやすいため，励起により重点をおいていることを表す．

W_{prop}, W_{excite} の値は，テスト生成時に各フリップフロップ集合の状態変化の優先度を決定するために用いられる．

対象故障の検出に影響するフリップフロップの重要度を判断するために，各フリップフロップ FF_i に対して，下記に示す $FF_{score}(i)$ を計算する．

$FF_{score}(i)$ は，フリップフロップと対象故障の位置関係と W_{prop}, W_{excite} の値を用いて計算される．

図 6 に，対象故障とフリップフロップとの位置関係の例を示す．

図 6 内の f は対象故障を表し，故障の入力部分回路を $\mathrm{T}_{excite}(f)$ と表す．また，故障の影響の伝搬部分回路を $\mathrm{T}_{prop}(f)$ と表す．図 6 の場合 $FF1$, $FF2$ は，伝搬部分回路に存在するため，$\mathrm{T}_{prop}(f)$ に含まれる．$FF3$ は，入力部分回路に存在するため，$\mathrm{T}_{excite}(f)$ に含まれる．

故障を検出するための各フリップフロップの重要度 FF_{score} は，次式により定義される．

$$FF_{score}(i) = \sum_{k=1}^{N} (P(f_k)W_{prop}(f_k) + E(f_k)W_{excite}(f_k)) \qquad (6)$$

式 (6) の N は総故障数，$P(f_k)$ はフリップフロップ FFi が対象故障 f_k の伝搬部分回路 $\mathrm{T}_{prop}(f_k)$ に含まれる場合に 1，そうでない場合に 0 とする．また，$E(f_k)$ は FF_i が対象故障 f_k の入力部分回路 $T_{excite}(f_k)$ に含まれる場合に 1，そうでない場合に 0 とする．

図 7 に FF_{score} の計算例を示す．図 7 の入力部分回路を $T_{excite}(fa)$, $T_{excite}(fb)$ と表し，伝搬部分回路を $T_{prop}(fa)$, $T_{prop}(fb)$ と表す．また，fa に関係する部分を実線，fb に関係する部分を破線で表す．図 7 より，FF_1 は $T_{excite}(fa)$, $T_{excite}(fb)$, $T_{prop}(fa)$, $T_{prop}(fb)$ のいずれにも含まれていないため $FF_{score}(1) = 0$ となる．$FF2$ は，fa の入力部分回路 $T_{excite}(fa)$ 内に存在し，かつ，fb の伝搬部分回路 $T_{prop}(fb)$ にも含まれている．したがって，$FF_{score}(2) = W_{excite}(fa) + W_{prop}(fb) = 19 + 10 = 29$ となる．同様に，$FF_{score}(3) = 9 + 10 = 19$, $FF_{score}(4) = 9 + 10 = 19$, $FF_{score}(5) = 9 + 18 = 27$, $FF_{score}(6) = 0$ となる．

文献 [2] においては，W_{prop}, W_{excite} の値を故障の位置に関わらず一定としているが，本研究における FF_{score} は，伝搬経

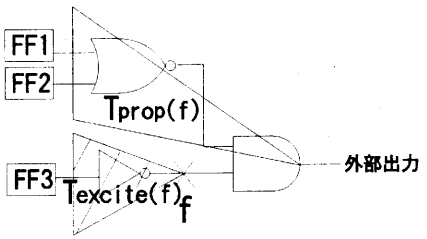

図 6　対象故障とフリップフロップの位置関係例

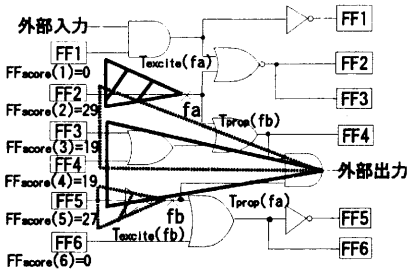

図 7　FF_{score} の計算例

G1		G2		G3	
FF2	FF5	FF3	FF4	FF1	FF6
W1:3		W2:2		W3:1	

図 8　対象故障を考慮する回路構造に基づく状態集合分割例

路の多い故障に影響するフリップフロップのうち，励起に必要な入力部分回路内のフリップフロップに対して，大きい値を割り当てるよう定義されている．図 7 の例においては，$FF2$ の方が $FF3$ よりも大きい FF_{score} が割り当てられている．これにより，fa の入力部分回路に属する $FF2$ の値の変化を優先させることが可能となる．

FF_{score} が小さいフリップフロップの値が変化する状態遷移よりも，FF_{score} が大きいフリップフロップの値が変化する状態遷移を優先することで，未検出故障に影響するフリップフロップの値の変化が多くなり，故障検出に必要となるテストベクトルの現れる確率が高くなると考えられる．

以下に，対象故障を考慮する回路構造に基づく状態集合分割法の手順を示す．本手法では，FF_{score} の値が大きいフリップフロップから順に，各集合でフリップフロップ数が等しくなるように分割を行う．また，FF_{score} の値が大きい集合の状態遷移を優先させるため，大きい FF_{score} を分配した集合に，大きい重みを割り当てる．

（1） 式 (4)，式 (5) を用いて各故障の W_{prop}，W_{excite} を計算

（2） フリップフロップ i と対象故障との位置関係により，式 (6) を用いて $FF_{score}(i)$ を計算

（3） FF_{score} の値が大きい順に，フリップフロップを G_1〜G_5 に 5 等分する

（4） 集合 G_1, G_2, G_3, G_4, G_5 に重み W_1, W_2, W_3, W_4, W_5 を割り当て

ここで，割り当てた重みは $W_1 > W_2 > W_3 > W_4 > W_5$ である．これにより，未検出故障に対する状態分割を行う．図 7 で計算した FF_{score} の値に従い，フリップフロップを 3 等分した例を図 8 に示す．

図 7，図 8 に示すように，FF_{score} の値が大きい集合に大きい重みを割り当てる．集合 G_1 に，FF_{score} の大きい $FF_{score}(2) = 29$, $FF_{score}(5) = 27$ の $FF2$, $FF5$ を配分し，集合 G_2 に，FF_{score} の値が $FF_{score}(3) = 19$，$FF_{score}(4) = 19$ の $FF3$, $FF4$ を配分，集合 G_3 に，FF_{score} の値が小さい $FF_{score}(1) = 0$, $FF_{score}(6) = 0$ の $FF1$, $FF6$ を配分する．また重み W_j は，集合 G_1 に一番大きい重み $W_1 = 3$ を割り当て，集合 G_2 に $W_2 = 2$，G_3 に $W_3 = 1$ を割り当てる．

4. テスト生成手法

3 章で述べた状態集合分割法を論理シミュレーションを用いるテスト生成へ適用する．テスト生成時に，候補ベクトルの選択に用いる評価式を式 (7) に示す．候補ベクトルの適応度 $opt(V_l)$ の値は，各状態集合に割り当てた重みにより決定される．

$$opt(V_l) = \sum_{j=1}^{N} 2^{W_j} \alpha_j(V_l) \qquad (7)$$

式 (7) の N は総集合数，W_j は集合 G_j に割り当てた重みを表し，$\alpha_j(V_l)$ は集合 G_j の状態が V_l により既到達状態へ遷移する場合に 0，未到達状態へ遷移する場合に 1 となる．また，候補ベクトル生成法として遺伝的アルゴリズム [4] を用いる．

以下に，テスト生成手順を示す．

（1） フリップフロップ制御容易性分割を用いてフリップフロップを集合 $G_1, G_2, G_3, G4, G_5$ に分割

（2） 乱数を用いて候補ベクトル V を 100 個生成

（3） $V_l\{l = 1\cdots100\}$ による論理シミュレーションを行い，遷移先の状態 $S_m\{1\cdots100\}$ を集合ごとに記憶

（4） 式 (7) を用いて opt を計算し，opt が最大となるベクトル V_n をテストベクトルとして採用

- opt が同値の場合，最初に opt が最大となった候補ベクトルをテストベクトルとして採用
- $opt = 0$ の場合，直前の状態と異なる状態へ遷移可能なベクトルをテストベクトルとして採用

（5） V_n の遷移先の状態を集合 $G_j\{j = 1\cdots5\}$ ごとに記憶

- フリップフロップ制御容易性分割によるテスト生成の場合 7 へ
- 対象故障との構造による分割によるテスト生成で，再分割を行った場合 7 へ

（6） 採用ベクトル V_n を用いて故障シミュレーションを行い，100 ベクトル連続で新故障を検出不可能であった場合，この時点での未検出故障を対象故障として構造による分割法を用いてフリップフロップを再分割

（7） テストベクトルを 100000 個生成すれば終了，達していなければ 8 へ

（8） 突然変異，一点交叉を用いて候補ベクトル V を 100 個生成し，3 へ

遺伝的アルゴリズムにおいて用いたパラメーターは，一点交叉率 0.8，突然変異率 0.04 である．

5. 実験結果

提案手法を C 言語で記述し，PC-UNIX Celeron1.7GHz 上で実行した結果を示す．$ISCAS89$ ベンチマーク回路に対して，フリップフロップ制御容易性分割法と対象故障を考慮する回路構造による分割法を適用しテスト生成を行った．得られたテスト生成結果を表 1 に示す．

表 1　提案手法と文献 [2] によるテスト生成結果

回路名	全故障数	フリップフロップ制御容易性分割			構造による分割			文献 [2] に基づく分割		
		検出数	系列長	CPU(s)	検出数	系列長	CPU(s)	検出数	系列長	CPU(s)
S641	467	407	31,259	297	407	20,532	279	408	11,735	71
S1196	1,242	1,232	51,074	1,213	1,233	99,546	1,114	1,227	83,100	1,156
S1238	1,355	1,274	83,239	1,160	1,274	84,060	1,273	1,275	98,070	1,581
S1488	1,486	1,277	95,085	374	1,294	96,159	398	893	75,000	350
S1494	1,506	1,256	91,228	345	1,290	89,305	315	930	99,828	300
S5378	4,603	3,265	71,775	20,401	3,346	97,802	2,704	3,302	71,775	2,986

表 2　従来法によるテスト生成結果との比較

回路名	全故障数	構造による分割					文献 [1]			文献 [6]		
		検出数	検出率	系列長	圧縮後	CPU(s)	検出率	系列長	CPU(s)	検出数	系列長	CPU(s)
S641	467	407	87.2	20532	117	279	85.6	935	24			
S1196	1,242	1,233	99.3	99,546	232	1,114	95.3	3,137	94	1,239	574	89
S1238	1,355	1,274	94.0	84,060	253	1,273	90.0	3,053	147			
S1488	1,486	1,294	87.1	96,159	272	398						
S1494	1,506	1,290	85.7	89,305	279	315				1,453	540	456
S5378	4,603	3,346	72.7	97,802	617	2,704	68.0	2,769	3,934	3,639	11,571	136,080

表 1 の全故障数の欄は縮退故障数を表す．フリップフロップ制御容易性分割及び，構造による分割における系列長の欄は 100000 個のテストベクトルによる故障シミュレーションの際，最後に故障を検出したベクトル数を表している．また，CPU の欄は計算時間を表す．文献 [2] に基づく分割の欄は，文献 [2] との比較のため，W_{excite}, W_{prop} の値を故障の位置に関わらず，それぞれ 5,10 とし，状態集合分割を行った結果を表している．文献 [2] での候補ベクトル生成法の再現が困難であったため，本実験においては，4 章で述べた候補ベクトル生成を行っている．文献 [2] に基づいた手法と比較すると，$s641$，$s1238$ を除く回路ではより多くの故障を検出することが可能であった．また，$s1238$ では短い系列長でほとんど同数の故障を検出することが可能であった．従来法との比較のため，ランダムパターンを用いた論理シミュレーションによるテスト生成の結果を表 2 に示す．文献 [1] で使用した計算機は，SUN SPARC station IPX であり，文献 [6] で使用した計算機は，HP 9000 J200 with 256 MB RAM である．文献 [1] の結果については，検出故障数が記載されていないため検出率のみを示す．文献 [6] の結果については，対象故障数が記載されていないため検出故障数のみを示す．結果を文献 [1] と比較すると，より多くの故障を検出することが可能であった．また，$s1238$ を除く回路ではフリップフロップ制御容易性分割法よりも，対象故障との構造による分割法の方が高い故障検出率を達成することができた．また，検出故障数が同じ場合でも短いテスト系列長で故障を検出することが可能であった．文献 [6] と比較すると，検出故障数では本手法が下まわっているが，$s1494$ と $s5378$ では，短い計算時間でテストベクトルを生成することが可能であった．しかし，文献 [1] よりも高い故障検出率を達成している場合でもテストベクトル数が非常に多くなってしまっている．そのため文献 [5] を基にしたテスト圧縮プログラムを用い，テストベクトル数の削減を行った．表 2 の圧縮前の系列長の欄は，回路構造による分割法を用い，生成したテストベクトルである．表 2 に示すように，テスト圧縮を用いることによりテストベクトル数を大幅に削減することが可能であった．

6. ま と め

本論文では，状態集合分割法として，フリップフロップ制御容易性分割と，対象故障を考慮する回路構造による分割を行い，ベンチマーク回路に適用した実験結果を述べた．結果より，フリップフロップ制御容易性分割法の適用を行い，さらに対象故障を考慮する構造による分割法を併用することで高い故障検出率を達成することが可能であることが分かった．今後テスト生成プログラムを改良し，実行時間の短縮を行う予定である．

謝　辞

本研究で使用したテスト圧縮プログラムは，愛媛大学の樋上喜信先生に提供していただいた．ここに感謝の意を表します．

文　献

[1] 高松雄三，児玉剛，東功：“ランダム入力による並列故障シミュレーションを利用した順序回路のテスト生成”，信学論 (D-I)，J77-D-I, 12, pp. 803-811, Dec. 1994.

[2] S. Sheng, M.S. Hsiao, "Efficient Sequential Test Generation Based on Logic Simulation," IEEE Design & Test of Computers, vol. 19, no. 5, pp. 56-64, 2002.

[3] I. Pomeranz, S.M. Reddy, "LOCSTEP:A Logic-Simulation-Based Test Generation Procedure,"IEEE Trans.CAD of Integrated Circuits and Systems, vol. 16, no. 5, pp. 544-554, May 1997.

[4] L. Davis (編)，嘉数侑昇，三上貞芳，皆川雅章，川上敬，高取則彦，鈴木恵二 (訳)，遺伝アルゴリズムハンドブック，森北出版社，1994.

[5] R. Guo et.al, "On Speed-up Vector Restoration Based Static Compaction of Test Sequences for Sequential Circuits," Proc. Asian Test Symposium, pp.467-471, 1998.

[6] M.H. Hsiao, E.M. Rudnick, and J.H. Patel, "Sequential Circuit Test Generation Using Dynamic state Traversal," Proc. European Design & Test Conference, pp.22-28, 1997