7-9

テスト圧縮効率化のためのテストパターン再生成の考察

日大生産工(院)　○山崎 達也　　日大生産工　細川 利典
九大　吉村 正義　　明大　山崎 浩二

1. はじめに

近年，半導体微細化技術の進歩に伴い，大規模集積回路（Large Scale Integration: LSI）の大規模化によるテストコストの増加が問題となっている[1]. テストコストはテストパターン数と比例関係にあるため，テストパターン数を削減することにより，テストコストの削減が期待できる．

小規模の回路では，ほぼ最小のテスト集合を得るアルゴリズム[2][7]が提案されている．しかしながら，大規模回路では最小のテスト集合得るための計算量が多く，現実的な計算時間での適用は困難である．そこで大規模回路に適用可能な手法として，ドントケア故障シミュレーションを用いたテスト圧縮法[3]が提案されている．静的圧縮法として，文献[9]で提案されているようにドントケア抽出[4]を用いて，文献[3]の方法で生成されたテストパターン中のドントケア率を増加させた後で，極小解圧縮[5]と 2 重検出法アプローチを採用する．文献[5]において，テストパターンのドントケアに基づく静的圧縮における曲紹介圧縮の貪欲解と厳密解がほぼ一致することが報告されている．極小解圧縮を行ったテストパターンに対し，ドントケアの割当てを行い 2 重検出法[7]を用いることにより，さらに不必要なテストパターンを削除する．

また，静的圧縮の圧縮率は，生成されたテストパターンに依存する[5]. 本論文では更なる圧縮率の向上を目指すため，テスト圧縮の効率を下げるテストパターンを定義し，そのテストパターンでしか検出されない故障に対して，テスト生成アルゴリズムを変更して，テストパターンを再生成する．

2. ドントケア故障シミュレーションを用いたテストパターン圧縮

本章ではドントケア故障シミュレーションを用いたテストパターン圧縮手法[3]を説明する．故障シミュレーションとは，生成されたテストパターンに対し，どの故障が検出されているかを解析するものである．テストパターン生成時に故障シミュレーションを行なうことで，テストパターン生成回数が削減し，テストパターン生成数が減ることにより，テストパターンを圧縮できる．通常，テストパターンは 0，1 の 2 値で故障シミュレーションを実行するのに対し，ドントケア故障シミュレーションでは，0，1，X（ドントケア）の 3 値のテストパターンを用いる．図 1 にドントケア故障シミュレーションによるテスト圧縮で用いる故障表の例を示す．故障表は各故障に対して検出フラグを備えている．故障表から未検出故障を 1 つ取り出し，その故障に対してテストパターンの生成を行なう．また，一度テスト生成を試みた故障に対しては再度テスト生成されることはないものとする．

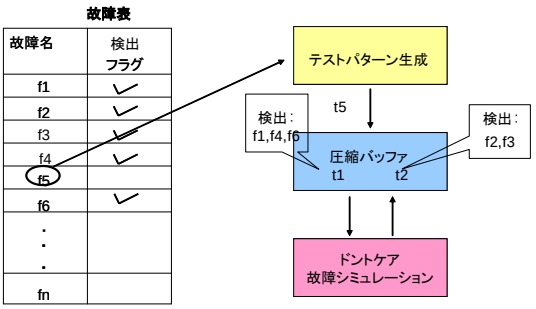

図 1. 故障表とその更新例

検出フラグとは故障が検出されたか否かを判断するためのフラグであり，生成された 3 値のテストパターンで故障シミュレーションを実行した際，そのテストパターンによって検出される故障の検出フラグにチェックされる．検出フラグにチェックがついている故障は，生成されたテストパターンで検出可能なため，以降のテスト生成や故障シミュレーションの対象となる故障集合から外す．

図 1 の故障表の更新について説明する．まず未検出故障 f1 に対してテストパターン t1 が生成される．テストパターン t1 を圧縮バッファに格納させる時に，故障 t1 に対しドントケア故障シミュレーションを実行する．ここで故障 f1，f4，f6 を検出できたとすると，この情報から故障表の故障 f1，f4，f6 に対する検出フラグをチェックし，故障テーブルを更新する．次に未検出故障 f2 に対してテストパターン t2 が生成される．テストパターン t2 に対しドントケア故障シミュレー

On The Consideration of Test Pattern Regeneration for Test Compaction

Tatsuya YAMAZAKI, Toshinori HOSOKAWA
Masayoshi YOSHIMURA, and Koji YAMAZAKI

ションを実行する．ここで故障 f2，f3 を検出できたとすると，この情報から故障表の故障 f2，f3 に対する検出フラグをチェックし，故障テーブルを更新する．次に未検出故障 f5 に対してテスト生成を行なう．この時故障 f3，f4 は先に生成したテストパターン t1，t2 により検出されているので，テスト生成の対象にならない．

3. 極小解圧縮

本章では極小解圧縮について説明する．極小解圧縮は，貪欲法でテストパターンを圧縮し，極小解を求めることで，最小個のテストパターン数とほぼ一致する値を得る圧縮法である．

本論文では極小解圧縮を解くために，頂点彩色問題[5]を解く貪欲法のアルゴリズムを採用している．テスト圧縮での頂点彩色問題を説明する．まず，頂点をテストパターン，圧縮可能なテストパターンを辺で結ぶことによりテストパターンの圧縮可能性を無向グラフとして表現することができる．頂点彩色問題はそのグラフに対する補グラフを作成し，グラフの隣接する頂点が異なる色になるように，頂点に色を塗るために必要な最小の色数を求める問題である．本論文では頂点彩色問題の貪欲法アルゴリズムの一つである Dsatur [6]を用いる．

4. 2 重検出法

2 重検出法とは, 各テストパターンの必須故障情報[]を基にしたテストパターン選択法である．必須故障とは，テストパターン集合内で，1 つのテストパターンでしか検出できない故障である．2 重検出法を用いることにより，極小テスト集合[5]を，極小テスト集合のシミュレーション順位を最優先することにより，冗長なテストパターンを削除することができる．

表 1 に 2 重検出法に用いる 2 重検出法故障表の例を示す．2 重検出法故障表は各テストパターンに対して検出可能な故障とそのテストパターンの必須故障数を備えている．表 1 の例では，t1 の検出可能な故障は f1，f2 である．また，f2 と f5 は必須故障であるため，それらを検出できるテストパターンである t1 と t4 の必須故障数はそれぞれ 1 である．t1，t4 の極小テスト集合でまず故障シミュレーションを実行することにより f1，f2，f4，f5 の故障を検出することができる．f3 を選択したとすると，t3 を削除することができる．

表 1. 2 重検出法故障表

テストパターン	検出故障	必須故障数
t1	f1,f2	1
t2	f1,f3	0
t3	f3,f4	0
t4	f4,f5	1

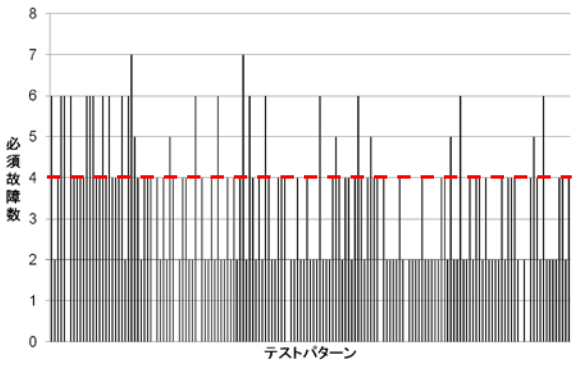

図 2. 各テストパターンに対する必須故障数

5. テスト圧縮の効率を下げるテストパターン

本論文ではテスト圧縮に極小解圧縮と 2 重検出法を用いたが，この 2 つの手法は静的圧縮であるので，与えられたテスト集合に圧縮率が依存してしまう．そこで圧縮率を下げるテストパターンを定義し，そのテストパターンを圧縮しやすいテストパターンとして再生成することで圧縮率の向上をはかる．

本論文では圧縮率を下げるテストパターンを，検出する必須故障数が閾値 n 以下であるテストパターンと定義する．閾値 n は整数である．必須故障が少ないテストパターンは，削除しても故障検出率への影響が少なく，その必須故障を検出するテストパターンを再生成したときに，削除した複数のテストパターンが検出すべき必須故障を検出可能な一つのテストパターンを生成する可能性があると考えられるためである．図 2 はテストパターンに対する必須故障数を表すグラフである．横軸が各テストパターンで，縦軸が各テストパターンに対する必須故障数を表す．各テストパターンの必須故障数の平均値を計算し，その値を基に閾値を設定する．閾値に満たないテスト集合でのみ検出される故障に対してテストパターンを再生成し，閾値に満たないテストパターンと交換する．この処理により，圧縮率を下げる可能性があるテストパターンが再生成され，圧縮率を向上できる可能性がある．図 2 では閾値は 4 となり，必須故障数が 4 未満のテストパターンは再生成される．

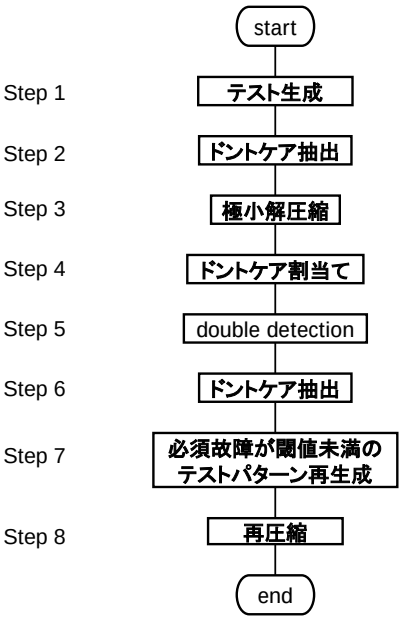

図 3. 全体のフローチャート

6. テスト圧縮アルゴリズム

図 3 にテスト圧縮全体のアルゴリズムを示す.

(step1)

テスト生成を行う．テスト生成アルゴリズムは正当化よりも先に、故障の影響を単一経路を通って外部出力まで伝搬させる SPOP アルゴリズムを用いる．この step でドントケア故障シミュレーションを実行している.

(step2)

生成されたテスト集合に対し．ドントケア抽出を行う．(step3)

極小解圧縮を実行する.

(step4)

テストパターン圧縮されたテストパターン中のドントケアにランダムに 0 または 1 を割当てる.

(step5)

2 重検出法を実行する.

(step6)

ドントケア抽出を行う.

(step7)

必須故障の検出数が閾値に満たないテスト集合のみで検出される故障に対しテスト生成を行なう．このテスト生成法において step1 と同様のアルゴリズムを用いてしまうと，同じテストパターンが生成されてしまうので，この step のテスト生成アルゴリズムは step1 と異なる方法で実施する．本論文では，FAN アルゴリズムを適用する.

(step8)

閾値以上のテストパターン集合と，再生成したテストパターン集合を併せて極小解圧縮，2 重検出法を実行する.

7. 実験結果

テスト圧縮の効率を下げるテストパターンを必須故障数の少ないテストパターンとし，そのテストパターンに代わるテストパターンを再生成しテスト圧縮して，テストパターンの削減率の評価を行った．実験では ISCAS'89 ベンチマーク回路を用いた．表 1 に圧縮の効率を下げるテストパターンを必須故障数の少ないテストパターンと仮定し，そのテストパターンに代わるテストパターンを再生成しテスト圧縮し，テストパターン再生成前とテストパターン再生成後の削減率の効果を比較する．表 2 おいて「circuit」は回路名である．「no com」テスト対象回路にテスト生成を行なった（図 3 のフローチャート step1 まで実行）テストパターン数,「first com」は「no com」のテストパターンに極小解圧縮と 2 重検出法を実行した（図 3 のフローチャート step5 まで実行）テストパターン数,「second com」は「first com」閾値以下のテストパターンを再生成した後に圧縮したテストパターン数である．「re fault」は再生成した際のテスト対象故障数の割合を示しており，テストパターンを再生成時の対象故障数/テスト生成対象回路の代表故障数×100 で計算している．「under pat」は閾値以下のテストパターン数，つまり圧縮の効率を下げるテストパターンとしてテスト集合から削除されたテストパターンである．「re pat」は閾以下のテストパターンが削除されたために，検出できなくなった故障に対し再生成したテストパターン数である．「reduce」はテストパターンを再生成したことによるテストパターンの圧縮率を示しており，「second com」のテストパターン数「first com」のテストパターン数×100 で計算している．閾値は全ての回路において，4 を設定している.

表 2 から，圧縮の効率を下げるテストパターンを，必須故障数の少ないテストパターンと仮定し,再生成，圧縮することでテストパターン数が約 4%削減されており，圧縮の効率を下げるテストパターンを，必須故障数が少ないテストパターンとして再生成することが，テスト圧縮率の向上において有効であるとわかる．また表 2 から圧縮の効率を下げるテストパターンとして削除したテストパターン数より，再生成したテストパターン数が多いことが分かる．しかし，結果としてテストパターン数が削減されているので，再生成したテスト集合内のテストパターンが検出する必須故障の組合せが変化したことにより，テストパターンが圧縮の効率を上げるテストパターンに再生成されたことが分かる.

表 2. テストパターン数の比較

circuit	no com	first com	second com	re fault(%)	under pat	re pat	reduce(%)
s1423	537	44	41	1.65	4	10	7.31
s1488	303	106	103	7.13	54	85	2.91
s1494	302	106	101	16.9	53	89	4.95
s5378	1533	130	127	2.63	53	119	2.36
s9234	2037	161	158	2.32	28	61	1.89

8. おわりに

本論文では，テスト圧縮の向上を目指すために，圧縮の効率を下げるテストパターンを，必須故障数が少ないテストパターンとして，そのテストパターンの再生成，再圧縮した結果を示した．今後の研究の課題として,圧縮の効率を下げるテストパターンの再生成を，他のテストパターンとの圧縮を考えたテストパターンに変換することがあげられる．本論文のテストパターンの再生成は ATPG アルゴリズムを変えただけであったが，ATPG ツールを改良し 1 つの故障に対し，複数のテストパターン生成を行うことで，圧縮率を向上させる可能性を持つテストパターンを選択することが考えられる．また，今後の研究課題としてテストパターンを再生成する閾値を変更し，最適な閾値を判断することや，圧縮しづらいテストパターンの定義を，極小解圧縮の際に生じる最大独立点集合[5]にすることがあげられる．

参考文献

[1] Y.Sato, T.Ikeda, M.Nakao, and T.Nagumo, “Abist approach for very deepsub-micron (vdsm) defect, ”Proc. International Test Conference, pp. 283291, 2000.

[2] Y.Matsunaga, “MINT-An exact algorithm for finding minimum test set”, IEICE Trans. Fundamentals vol.E76-A, pp1652-1658 ,1993

[3] 秋山祐介, “ドントケア故障シミュレーションを用いた動的テスト圧縮の効率化”, 平成 18 年度日本大学生産工学部学術講演会数理情報部会講演概要集 ,2006

[4]K. Miyase, S. Kajihara “XID: Don’t Care Identification of Test Patterns for Combinational Circuits,” IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, Vol. 23, No. 2, pp. 321-326,Fed. 2004

[5] 八木澤圭, “テストパターンの静的圧縮における厳密解と貪欲解の比較” IEICE Technical Report DC2 007-79 ,2008

[6] D.Brelaz, “New methods to color the vertices of a graph”,Communications of the ACM, 22, p p.251-256,1979

[7]Seiji Kajihara, lrith Pomeranz, Kozo Kinoshita and Sudhakar M. Reddy “Cost-Effective Generation of Minimal Test Sets for Stuck-at Faults in Combinational Logic Circuits”, 30th ACM/IEEE Design Automation Conference, pp102-106,1993

[8] Ilker Hamzaoglu, Janak H. Patel, “Test Set Compaction Algorithms for Combinational Circuits” ,IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMSVOL.19, NO. 8, 2000

[9]谷口謙二郎，宮瀬紘平，梶原誠司，ポメランツ　イリス，レディ　スダーカ，“符号化技術を用いた多重スキャン回路のテストデータ量削減について”, 情報処理学会　研究報告，2002-SLDM-107，pp85-90，2002