ISSN 2186-5647

2-77

独立故障グラフを用いたテストパターン数下界の評価

日大生産工　○日下部　建斗　日大生産工(院)　山崎　紘史
日大生産工　細川　利典　九大　吉村　正義

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 Integrated circuits: LSI）が大規模化・複雑化している．それに伴い,テストパターン数が増加し,テストコストが増大するという問題が発生している[1]．テストコストはテストパターン数と比例関係にあり,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．

小規模回路では,ほぼ最小のテスト集合を得る静的圧縮と動的圧縮を適用したアルゴリズム[2]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を得るための計算量が大きく,現実的な計算時間での適用は困難である．

大規模回路に適用可能な手法として，文献[3]では,自動テストパターン生成(Automatic Test Pattern Generation ： ATPG)が生成した初期テスト集合に対しドントケア抽出[4]，頂点彩色問題を用いた極小解圧縮[5][6]，2 重検出法[2]を繰り返し行う静的圧縮手法が提案されている.しかしながら,文献[3]の手法では,初期テスト集合に対して静的圧縮を行うため,テスト圧縮効率が初期テスト集合に依存する.したがって,初期テスト集合中にテスト圧縮に非効率的なテストパターンが存在したとき,テストパターン数削減の妨げになる可能性があり,初期テスト集合に対する最終テスト集合のテストパターン数の削減率が小さくなる.よって,初期テスト集合中のテスト圧縮に非効率的なテストパターンを削除し,テスト圧縮に効率的なテストパターンを再生成するか,またはテスト圧縮に効率的な初期テスト集合を生成することにより,最終テスト集合のテストパターン数が削減されると考えられる.

文献[7]で提案されたように初期テスト集合を生成する時の動的テスト圧縮において,1 次故障や 2 次故障をテストパターン中のケアビット箇所に基づいて選択することが重要であると考える．しかしながら，テスト圧縮の研究を行う上で,まず評価の対象となるベンチマーク回路のテストパターン数の下界を明らかにし,定量的評価の目標として設定する必要がある．文献[2]では ISCAS’85，ISCAS’89 ベンチマーク回路のテストパターン数の下界を,独立故障集合[2]を用いて求めている.本論文では,文献[2]と同様に独立故障集合を用いて ITC’99 ベンチマーク回路のテストパターン数の下界を求め,テスト圧縮の研究を行うための定量的評価の目標を定める.故障モデルは単一縮退故障を対象とする．

2. 独立故障集合

独立故障集合とは,同一のテストパターンで検出することが不可能な故障の集合である.独立故障集合サイズが大きいものから順にテストの生成の目標故障として選択することで,より小さいテスト集合が得られると考えられる．

また,独立故障集合の最大サイズを求めることによってテスト対象回路のテストパターン数の下界を求めることができる.テストパターン数の下界とは,テスト対象回路の検出対象故障をすべて検出可能なテスト集合の最小数以下の数である．つまり，テストパターン数の下界を求めることで,テスト圧縮の研究を行う上での定量的評価の目標を定めることができる．

本論文では，3 章で示すように同時検出不可能故障グラフをクリーク分割することによって，独立故障集合を求めている．

An Evaluation of Lower Bound for the Number of Test Patterns Using Independent Fault Graphs

Kent KUSAKABE and Hiroshi YAMAZAKI and Toshinori HOSOKAWA and Masayoshi YOSHIMURA

3. 独立故障集合生成

3.1 同時検出不可能故障グラフ

同時検出不可能故障グラフとは，各故障を頂点として，同一のテストパターンで検出することが不可能な故障を辺で接続したものである．例えば，あるテスト対象回路の故障 f_1 と故障 f_2 を考える．それぞれの故障を検出するために必要な信号線値割当て(必須割当て)を求める．図 1 に示すように，故障 f_1 の検出には信号線 b, e に 0 を割当て，信号線 g に 1 を割当て，f_2 は信号線 c, d に 0 を割当て，信号線 a, e, f に 1 を割当てる必要があると仮定する．それぞれの必須割当てを比較すると，信号線 e で故障 f_1 と故障 f_2 の割当て要求が衝突している．よって，同一のテストパターンで検出することが不可能な故障として同時検出不可能故障グラフの頂点 f_1 と f_2 の間に辺を挿入する．

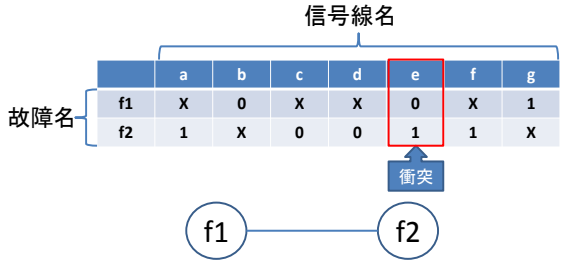

図 1. 同時検出不可能故障グラフ生成の例

図 2 に同時検出不可能故障グラフの例を示す．例として，あるテスト対象回路では検出対象故障が 5 個あり，それぞれの故障を f_1，f_2，f_3，f_4，f_5 とする．図 2 において，辺で接続された f_1 と f_5，f_1 と f_2，f_2 と f_3，f_2 と f_4，f_3 と f_4 はそれぞれ同時検出不可能故障であるということを表す．

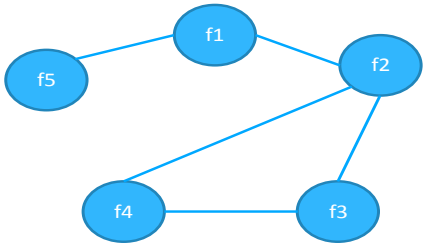

図 2. 同時検出不可能故障グラフの例

3.2 最大クリーク抽出

同時検出不可能故障グラフを生成した後，最大独立故障集合を求めるために最大クリーク抽出を行う．最大クリーク抽出とは，対象のグラフでクリーク分割を行ったときに，一つのクリーク内の頂点数であるクリークサイズが最大になるように，クリーク分割を行うことである．クリーク分割とは同時検出不可能故障グラフの各頂点を，部分的な完全グラフになるように分割することである．

図 3 に極大クリーク抽出アルゴリズムを示す[9]．

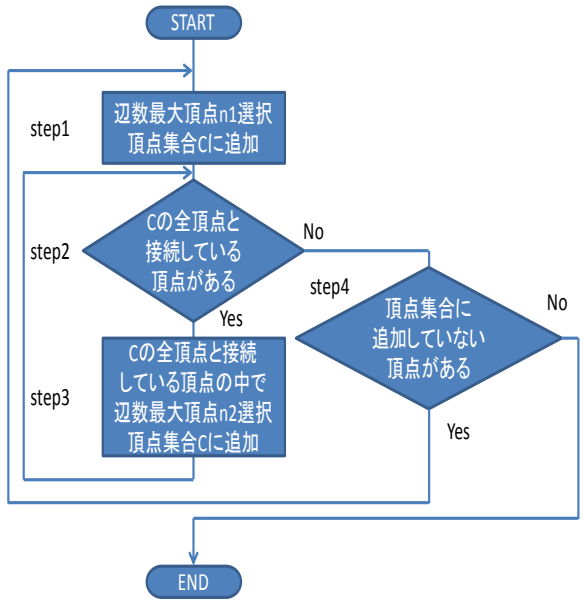

図 3. 極大クリーク抽出アルゴリズムのフローチャート

(step 1)

各頂点の辺の数を比較し，辺の数が最大の頂点を選択し n1 とする．n1 をクリーク頂点集合 C に追加し，n1 と n1 に接続する辺をグラフから削除する．

(step 2)

クリーク頂点集合 C に含まれるすべての頂点と接続している頂点が存在するか探索する．存在するならば step3 へ進み，存在しないならば step4 へ進む．

(step 3)

クリーク頂点集合 C に含まれるすべての頂点と接続している頂点の中で，辺の数が最大の頂点を選択し n2 とする．n2 をクリーク頂点集合 C に追加し，n2 と n2 に接続する辺をグラフから削除する．

(step 4)

C を Ci として保存し，C を空集合に初期化する．まだグラフ中に頂点が存在すれば step1 へ進む．それ以外の場合は処理を終了する．

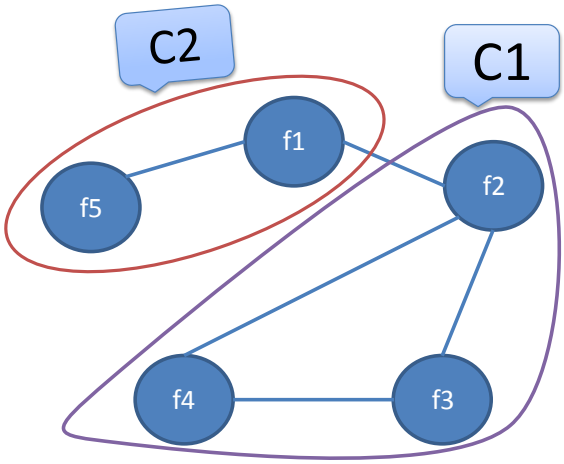

図 4. クリーク抽出した同時検出不可能故障グラフの例

図 4 に図 2 の同時検出不可能故障グラフを極大クリーク抽出した例を示す．クリーク C1（f2，f3，f4）とクリーク C2（f1，f5）の分割は解の一例である．この例では独立故障集合は 2 つで，最大独立故障集合サイズは C1 の 3，つまりテストパターン数の下界は 3 個と求めることができる．

3-3. 独立故障集合生成アルゴリズム

図 5 に独立故障集合生成アルゴリズムを示す．

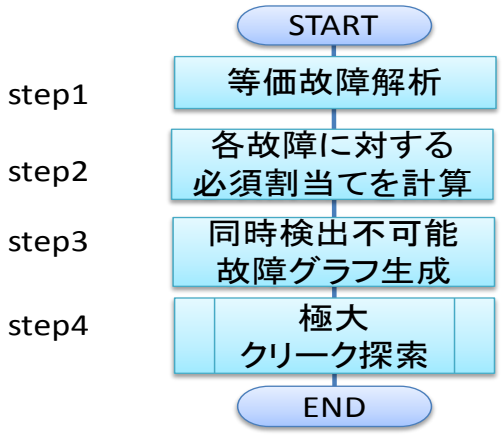

図 5. 独立故障集合生成アルゴリズムのフローチャート

(step 1)
テスト対象回路の等価故障解析を行い，代表故障集合を求める．
(step 2)
step1 で生成した各代表故障に対し，その故障を検出するために故障挿入[10]や一意活性化[10]を用いて含意操作[10]を行い必須割当てを算出し，各信号線に 0 又は 1 を割当てる．
(step 3)
各代表故障に対する信号線値の情報から同時検出不可能故障グラフを生成する．
(step 4)
step3 で生成した同時検出不可能故障グラフに対し極大クリーク抽出を行う．最大クリークのサイズが極大独立故障集合のサイズであり，テスト対象回路の下界が求まる．

4. 実験結果

独立故障グラフ生成を実装し、実験を行った．実験では ITC’99 ベンチマーク回路に対して必須割り当て計算に「故障挿入と含意操作」を用いたものと「故障挿入，含意操作，一意活性化」を用いたものを比較している．表 1 は独立故障グラフを用いた回路のテストパターン数の下界計算の実験結果である．表 1 において「回路名」はテストパターン数の下界を計算する対象の回路名である．「下界」は対象回路で独立故障グラフ生成を行い，極大独立故障集合サイズを計算し，テストパターン数の下界を計算した結果である．「辺の数」は対象回路で独立故障グラフ生成を行った際の独立故障の間にひかれた辺の数を示している．「枝密度(%)」とは対象回路で独立故障グラフ生成を行った際の独立故障の間にひかれた辺の割合を示している．表 1 から回路すべてにおいてテストパターン数の下界とするには，回路の規模に対して少ないものが存在する．これは必須割当ての部分においてまだ含意操作及び一意活性化しか適用していないため極大独立故障集合が正確に抽出できていないのではないかと考えられる．

	下界		辺の数		枝密度(%)	
回路名	含意操作判定	含意操作+一意活性化判定	含意操作判定	含意操作+一意活性化判定	含意操作判定	含意操作+一意活性化判定
b01	5	5	354/3192	360/3192	11.1	11.3
b02	3	4	212/1406	216/1406	15.1	15.4
b03	6	6	1920/31506	1928/31506	6.1	6.1
b04	6	11	13876/708122	19060/704760	2	2.7
b05	18	19	11520/949650	14632/947702	1.2	1.5
b06	6	6	526/7140	542/7140	7.4	7.6
b07	7	9	6006/268842	7632/268842	2.2	2.8
b08	8	8	1866/49952	2204/49952	3.7	4.4
b09	4	5	2008/53130	2176/53130	3.8	4.1
b10	9	9	2868/53130	3214/53130	5.4	6
b11	8	12	15530/551306	18868/549822	2.8	3.4
b12	33	33	43462/1749006	47408/1743720	2.5	2.7
b13	11	11	3950/229920	4212/229920	1.7	1.8
b14	27	29	961006/98992550	1038038/98972652	1	1
b15	45	70	2000512/87881250	2046512/86331972	2.3	2.4

表 1. 独立故障グラフを用いたテストパターン数の下界

5. おわりに

本論文ではテスト圧縮指向テスト生成を提案するための前段階として,独立故障グラフを用いてITC’99ベンチマーク回路のテストパターン数の下界を算出した.

今後の課題として,必須割当てを求める部分を改良し,より正確な回路のテストパターン数の下界を算出する.そして求めたテストパターン数の下界をテスト圧縮の研究を行うための定量的評価の目標として定め,テスト圧縮のさらなる考察を行う.最終的な目標として,独立故障グラフを利用したテスト圧縮指向テスト生成を提案することを目指す.

参考文献

1) Y.Sato, T.Ikeda, M.Nakao, and T.Nagumo, “A BIST Approach for Very Deep Sub-micron (VDSM) Defects, ”Proc. International Test Conference, pp. 283-291, 2000.

2) Seiji Kajihara, Irith Pomeranz, Kozo Kinoshita, “Cost-Effective Generation of Minimal Test Sets for Stuck-at Faults in Combinational Logic Circuits”, 30th ACM/IEEE Design Automation Conference, pp102-106,1993.

3) K. Miyase, S. Kajihara and Sudharkar M. Reddy, “A Method of Static Test Compaction Based on Don’t Care Identification”, IPSJ Journal, Vol.43, No.5, pp.1290-1293, May 2002.

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,Feb. 2004.

5) 八木澤圭, 山崎浩二, 細川利典, 玉木久夫, “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” 電子情報通信学会, DC,107, pp.77-82 , Feb. 2008.

6) D.Brelaz, “New Methods to Color the Vertices of a Graph” ,Communications of the ACM, Vol. 22, pp.251-256, Apr, 1979.

7) 山崎 達也, 細川 利典, 吉村 正義, 山崎 浩二, “テスト圧縮効率化のためのテスト生成に関する一考察” 第64回FTC研究会資料.

8) Michael H. Schulz, Erwin Trischler, and Thomas M. Sarfert “SOCRATES: A Highly Efficient Automatic Test Pattern Generation System”, IEEE Transactions on Computer-Aided Design, Vol.7, No.1, January 1988.

9)富田 悦次, 須谷 洋一, 東 貴紀, 高橋 真也, 若月 光夫, “単純でより高速な最大クリーク抽出アルゴリズム”pp1-4,情報処理学会研究報告

10) H. Fujiwara, Logic Testing and Design for Testability, The MIT Press, 1985.