ISSN 2186-5647

2-55

故障選択と圧縮バッファを利用した動的テスト圧縮法

日大生産工(院)　○楠山 友紀乃　　日大生産工　細川 利典
ルネサスマイクロシステム(株)　山崎 達也　　九大院　吉村 正義

1. はじめに

近年，半導体微細化技術の進歩によって，大規模集積回路(Large Scale Integrated circuits：LSI)が大規模化，複雑化している．さらに故障モデルの多様化も伴い，テストコストの増加が問題となっている[1]．LSI のテストコストは，テストに使用するテストパターン数と比例関係にある．そのため，テスト品質を落とさずにテストパターン数を削減するテスト圧縮[2][3]を用いることにより，テストコストの削減が期待できる．

テスト圧縮手法は，2 つのタイプに分類できる．テスト生成時にテスト集合の中のテストパターン数の削減を考慮しながら生成する動的圧縮[2][3]と，テスト生成後に生成されたテスト集合の中のテストパターン数を削減する静的圧縮[2][3]である．小規模の回路では，ほぼ最小解のテスト集合を得るアルゴリズム[4]が提案されている．しかし大規模回路では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．そこで大規模回路に適応しうる圧縮手法として，ドントケア故障シミュレーションを用いる動的圧縮手法[2]が提案されている．文献[2]のテスト圧縮アルゴリズムでは，動的圧縮に独立故障集合を用いたテスト対象故障選択，極大圧縮法，巡回後方追跡，2 重検出法[2]を用い，静的圧縮に独立故障集合を基に計算する Two-by-One 法を用いている．しかし従来法では，圧縮効率が不十分であるという課題を持つ．

また，文献[5]では，自動テストパターン生成ツール(Auto Test Pattern Generation：ATPG)が生成した初期テスト集合に対し，静的圧縮としてドントケア抽出[6]，テストパターンの併合[7]，2 重検出法を繰り返し行う手法が提案されている．しかし，文献[5]の圧縮手法は初期テスト集合に依存するので，初期テスト集合の内容によっては圧縮効率が悪い可能性がある．そこで，本論文では，文献[2]の独立故障集合を用いたテスト対象故障選択を用い，文献[5]の静的圧縮手法を，圧縮バッファ[8]の概念を用いて動的圧縮として実行するテスト圧縮アルゴリズムを提案する．動的圧縮として実行することで，初期テスト集合に依存せず，テスト集合の削減を目指す．

2. 圧縮バッファ内テスト圧縮手法

圧縮バッファ内テスト圧縮手法とは，テスト生成中に生成されたテストパターンが格納される圧縮バッファ内のテスト集合を，文献[5]で提案された手法を応用し圧縮するテスト圧縮手法である．文献[5]で提案された手法とは，テストパターンの併合，2 重検出法，ドントケア抽出を用いた圧縮手法である．圧縮バッファを用いることで，文献[5]で提案されている静的テスト圧縮手法を，動的に用いることができる．図 1 に圧縮バッファ内テスト圧縮アルゴリズムのフローチャートを示す．

(step1)
圧縮バッファ内テスト集合に対しテストパターン併合を実行．
(step2)
圧縮バッファ内テストパターンのドントケアに対し，0 または 1 をランダムに割り当てる．
(step3)
圧縮バッファ内テスト集合に対し 2 重検出法を実行
(step4)
圧縮バッファ内のテストパターンが検出する故障を指定したドントケア抽出を実行．

Dynamic Test Compression Technique using Selection Fault and Compression Buffer

Yukino KUSUYAMA　Tosinori HOSOKAWA Tatsuya YAMAZAKI and Masayoshi YOSHIMURA

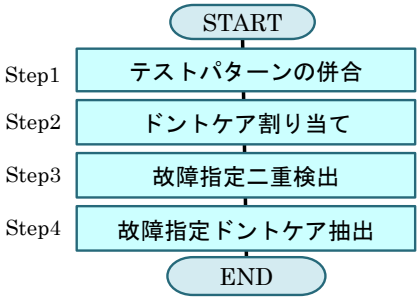

図 1. 圧縮バッファ内圧縮アルゴリズム

2-1. テストパターンの併合

テストパターンの併合とは，貪欲法でドントケアに基づくテストパターン圧縮問題[7]を解き，その極小解を基にテスト圧縮を実行することである．ドントケアに基づくテストパターン圧縮問題の貪欲解は，厳密解とほぼ同等であると報告されているため[7]，テストパターンの併合を適用するとほぼ最小のテスト圧縮解が得られると考えられる．

本論文ではテストパターンの併合を実行するために，ドントケアに基づくテストパターン圧縮問題を頂点彩色問題[7]に帰着させて解く貪欲法アルゴリズムを採用している．ドントケアに基づくテストパターン圧縮問題を説明する．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[9]を用いる．

2-2. 2 重検出法

2 重検出法[2]とは，各テストパターンの必須故障情報[2]を基にしたテストパターン圧縮法である．必須故障[2]とは，テスト集合内で，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，ドントケアに基づくテスト圧縮では削除できなかったテストパターンを削除できる．

3. 独立故障集合利用テスト対象故障選択

独立故障集合を用いたテスト対象故障選択ついて説明する．本手法では，テスト対象故障選択に，文献[2]の独立故障集合を用いたテスト対象故障選択を用いる．

表 1. 独立故障ペア例

	a	b	c	d	e	f	g	h
f1	X	X	0	X	0	X	0	0
f2	1	1	X	X	1	1	X	X

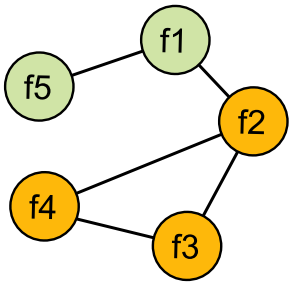

図 2. 独立故障ペアグラフ例

3-1. 独立故障集合

独立故障集合（Independent Faults Set）[2]とは，集合内のどの 2 つの故障のペアも同一のテストパターンで検出することができない故障の集合のことである．また，独立故障集合に含まれる故障数を独立故障集合サイズと呼ぶ．同一のテストパターンで検出することができない故障のペアを独立な故障ペアという．

表 1 に，独立故障ペアの例を示す．ある信号線 a～h からなる回路において，故障 f1 と f2 があるとする．それぞれの故障を検出するために，各信号線に必ず割り当てなければならない値は表 1 の通りであったとする．0・1 は故障を検出するためには，必ずその値を割り当てなければならない信号線．X は 0 または 1 どちらを割り当てても，故障を検出できる可能性がある信号線である．故障 f1 と f2 のペアは，信号線 e で衝突するので同一のテストパターンで検出することができない．よって，故障 f1 と f2 のペアは，独立故障ペアである．

3-2. 独立故障集合生成

独立故障集合の生成には，独立故障ペアグラフとクリーク分割を用いる．図 2 に，独立故障ペアグラフ例を示す．独立故障ペアグラフの各ノードは対象故障を示し，エッジで接続されている故障ペアは独立故障ペアであることを示す．この例では，5 つの対象故障 f1～f5 がある．独立故障ペアグラフをクリーク分割することで，独立故障集合を求めることができる．

独立故障集合内の故障数が最大のものを，最大独立故障集合という．独立故障ペアグラフを

最大クリーク分割することで, 最大独立故障集合を求めることができる. 図 2 の例では, 独立故障集合は【f2, f3, f4】【f1, f5】であり, 最大独立故障集合サイズは 3 である. 独立故障集合は集合内のどの 2 つの故障も同一のテストパターンで検出することができないので, 故障 f2 と f3, f2 と f4, f3 と f4 は同一テストパターンで検出不可能, つまり, テストパターン数の下界は 3 個である.

本手法では, 最大独立故障集合を求め, 独立故障集合サイズでソートを行い, 故障数が最大の独立故障集合に含まれる故障を優先的に選択することで, より下界に近いテストパターン生成を試みた.

4. テスト圧縮アルゴリズム

図 3 に, 本論文で提案したテスト圧縮アルゴリズムの全体のフローチャートを示す.

(step1)
すべてのテスト対象故障に対し最大独立故障集合を生成し, 独立故障集合サイズで降順ソートする.

(step2)
独立故障集合を基に故障を選択し, その故障に対しテスト生成を実行する.

(step3)
故障選択し生成されたテストパターンを圧縮バッファに格納する.

(step4)
圧縮バッファ内テスト圧縮を実行する.

(step5)
テスト対象故障がすべて検出済みなら step6 に進み, そうでないなら step2 に進む.

(step6)
生成されたテスト集合に対し静的圧縮を実行する. 静的圧縮アルゴリズムには文献[5]の手法を適用する.

5. 実験結果

本論文で提案した, 独立故障集合を用いたテスト対象故障選択を行う圧縮バッファ内テスト圧縮を実装し, 実験を行った. 実験では故障モデルを単一縮退故障とし, ISCAS'89 ベンチマーク回路を用いた. 表 2 は各回路に対してテスト生成とテスト圧縮を行い, 生成されたテストパターン数と実行時間の比較である.

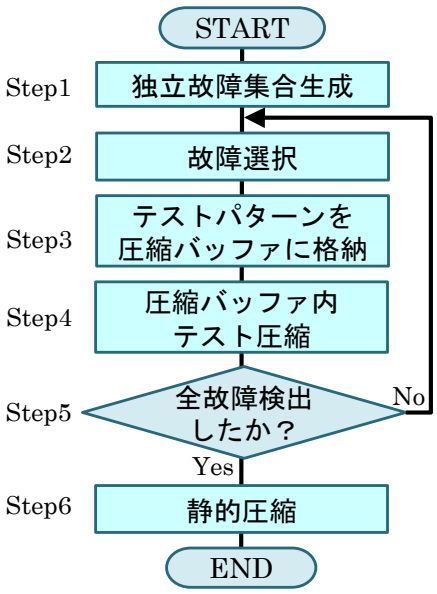

図 3. テスト圧縮アルゴリズム

表 2 において「回路名」はテストパターン集合を生成する対象の回路名である.「TP 数」は生成されたテストパターンの数,「実行時間」はテスト生成とテスト圧縮を行うのにかかった時間の合計である. また,「下界」は対象回路のテストパターン数の下界,「tetra」は synopsys 社のテスト生成ツール TetraMAX,「[2]」は文献[2]の独立故障集合を用いた故障選択を含む動的圧縮,「ランダム」と「IFS」はそれぞれ, 圧縮バッファ内テスト圧縮を, ランダムに故障選択を行った場合と独立故障集合を用いた故障選択を使用した場合の結果である.

表 2 から,「ランダム」「IFS」ともに「tetra」の結果と比較してテストパターン数が少なく, また一部の回路では下界と同じテストパターン数になっている. このことから, 圧縮バッファ内テスト圧縮が有効であることが分かる. また, ランダムよりも独立故障集合を用いた故障選択を使用した方が, テストパターン数が少なくなる回路がある. このことから, 独立故障集合を用いた故障選択を使用した方がより少ないテストパターン集合を生成することが可能であると考えられる. しかし, 文献[2]の結果と比較すると, あまりテストパターンの削減がみられない. さらに, ランダムよりも独立故障集合を用いた故障選択を使用した方が, 独立故障集合を求める分, 実行時間が増加している.

6. おわりに

本論文では, 独立故障集合を用いたテスト対象故障選択と圧縮バッファ内テスト圧縮を用いたテスト圧縮手法を提案し，実験を行った.

今後の課題として，独立故障集合を改善し, よりテストパターン数が少ない, テスト生成を目指す．また，テスト対象故障選択を改善し, よりテストパターン数が少なく, なおかつより高速な圧縮バッファ内テスト圧縮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) defect, ”Proc. International Test Conference, pp. 283291, 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）P.Goel and B.C.Rosales, “Test Generation and Dynamic Compaction of Tests,” Digest of papers 1979 Test Conf., pp189-192, 1979.

4）Y.Matsunaga, “MINT-An exact algorithm for finding minimum test sets,” IEICE Trans. Fundamentals vol. E76-A, pp1652-1658 (1993)

5）K. Miyase, S. Kajihara and Sudhakar 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.

6）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.

7）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, 107, pp.77-82 , Feb. 2008.

8）秋山祐介,細川利典,山崎浩二，“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 ,pp.95-98,2006.

9）D.Brelaz, “New methods to color the vertices of a graph” ,Communications of the ACM, 22, pp.251-256, 1979.

回路名	TP数					実行時間	
	下界	[2]	tetra	ランダム	IFS	ランダム	IFS
s208	27	27	30	28	28	1.40	1.62
s298	23	24	30	23	24	1.51	1.87
s344	13	16	18	13	14	0.94	1.32
s349	13	16	18	15	15	1.11	1.49
s382	25	25	28	26	25	2.36	2.44
s386	63	64	74	64	63	9.09	9.34
s400	24	24	30	25	24	2.48	2.79
s420	43	44	46	44	43	5.17	6.67
s444	24	25	29	25	25	3.68	4.16
s510	54	55	59	57	54	9.73	12.30
s526	49	50	61	49	49	7.77	9.71
s641	21	22	28	28	30	5.59	6.78
s713	21	26	33	27	24	11.06	12.28
s820	93	95	105	94	95	44.38	47.71
s832	92	96	105	94	96	52.81	53.28
s838	75	76	80	76	75	31.83	29.04
s953	73	77	139	77	73	38.09	36.83
s1196	105	120	139	118	120	116.66	130.60
s1238	115	132	152	127	127	189.94	212.50
s1423	16	33	38	32	32	22.67	27.60
s1488	100	101	118	102	102	93.83	114.04
s1494	100	101	114	100	100	96.58	112.74
s5378	92	105	126	105	103	816.32	1072
s9234	90	114	148	119	109	5761	7323

表 2. テスト圧縮を行ったテストパターン数と実行時間の比較