ISSN 2186-5647

2-69

テスト生成アルゴリズムにおける検出困難故障の評価

日大生産工(院)　○鈴木悠介　日大生産工(院)　山崎紘史　日大生産工　細川利典
九大院　吉村正義　明大　山崎浩二　読売理工　中尾教伸

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 Integtatied circuits:LSI)が大規模化・複雑化し，テスト生成時間の増加や故障検出効率の低下が問題となっている．この問題を解決するためには，スキャン設計[1]のようなテスト容易な設計技法で論理設計を行うとともに，さらに効率の良いテスト生成アルゴリズムを考案することが必要である．既存のテスト生成アルゴリズムは，経路活性化法[2-5]とSAT ベーステスト生成[6]に分類できる. 経路活性化法は，故障伝搬・正当化・バックトラックの処理を，故障影響が外部出力に伝搬し，かつ未正当化信号線が存在しなくなるまで繰り返し行う手法である．代表的なアルゴリズムとして FAN[2]，SOCRATES[3]，SPOP[4]，D アルゴリズム[5]提案されている．SAT ベーステスト生成は，論理回路を CNF に変換し，全ての変数を 1(真)，0(偽)に定めることで，式全体の値を1(テスト生成可能)とする割当てが存在するか判定する手法である．現在の問題として，マルチサイクル演算器などが多く使用されたレジスタ間のゲート段数が深い回路がある．この回路に対して，1 つのテスト生成アルゴリズムを適用するだけでは，現実的な時間で100%の故障検出効率を得ることは困難であると予測する．そのため，故障毎に得意なテスト生成アルゴリズムを適用する手法が有効だと考える．

本稿では，まず代表的なテスト生成アルゴリズムを実装し，検出困難故障数を算出する．不得意な故障の特徴を ITC'99 ベンチマーク回路を用いて，回路構造の面から考察する．

2.　経路活性化法

2.1　　故障伝搬

(1)パスセグメント活性化

図 1 にパスセグメント活性化のフローを示す.
(step1)故障箇所に故障情報を付加する.
(step2)値が一意に決定される信号線に割当てを行う.
(step3)故障影響が外部出力へ伝搬していれば，パスセグメント活性化の処理を終了し，正当化処理を行う.

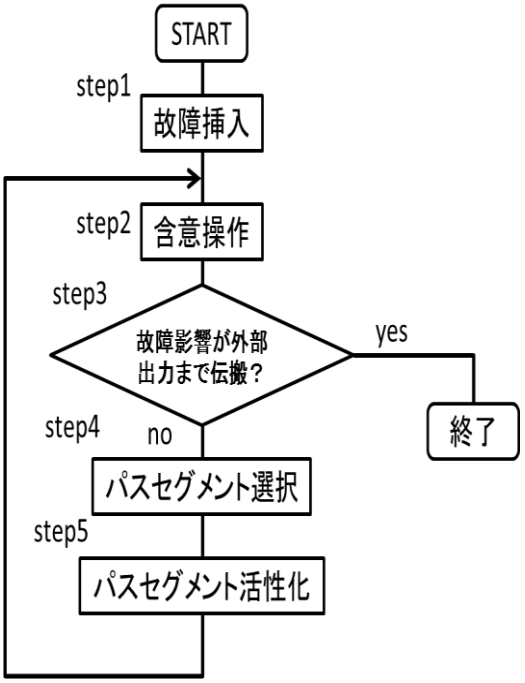


図 1 パスセグメント活性化フロー

伝搬していなければ step4 へ進む.
(step4)複数ある D フロンティア[1]の内、1 つ選択する.
(step5)選択した D フロンティアから分枝元信号線までの経路を活性化する.

図 2 にパスセグメント活性化の例を示す. 信号線 d, e は外部出力である．図 2 中の信号線 a が 1 縮退故障をしているとする．故障挿入と含意操作により，一意に値を割当てられる信号線値を決定する．まだ故障影響が外部出力に伝搬していないのでパスセグメントの選択を行う． D フロンティアの選択は決定木[1]を用いる. G1, G2 なのでこの例では G1 を選択する(図 3).
パスセグメント活性化により、次の分枝元信号線もしくは外部出力までのパスセグメントを活性化する. G1 の故障影響が伝搬していない信号線 b に AND ゲートの非制御値 1 が割当てられる．故障影響が外部出力 d に伝搬したため正当化の操作を行う.

Evaluation for hard-to detect faults of test generation algorithms

Yusuke SUZUKI, Toshinori HOSOKAWA, Hiroshi YAMAZAKI
Masayoshi YOSHIMURA, Kouji YAMAZAKI and Takanobu NAKAO

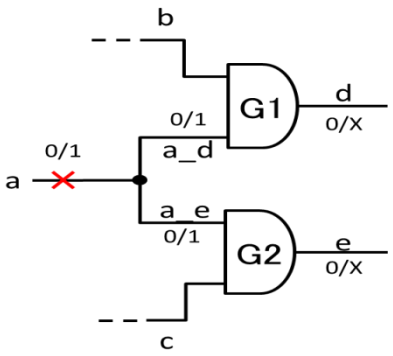


図 2 パスセグメント活性化例

図 3 D フロンティアの選択

(2)後方追跡

後方追跡は，回路内のすでに値が決定している信号線と矛盾しないように，目標とする信号線から信号線値が X(値が未決定)である入力に向かって，到達信号線に信号値を割当てる操作である．図 2 中では目標として，信号線 b に 1 を割当てるという目標で入力側に後方追跡を行う．本実験では，到達信号線として含意ノード[7]を用いる．含意ノードとは以下の条件を満たす信号線である．

・含意ノードの信号線は故障影響を受けない．

・信号線から論理値の割当て要求を少なくとも 1 つの外部入力まで一意に後方追跡できる経路が存在する．

今回は後方追跡を深さ優先と幅優先を用いて行う．

(1) 深さ優先

深さ優先探索は，初期目標に対して，外部入力まで後方追跡を行い，最初に発見した含意ノードが決定点をする．深さ優先探索は，決定点が論理段数の高い信号線になる場合がある(図 4)．

(2) 幅優先

幅優先探索は，後方追跡のアルゴリズムは多重後方追跡[2]と似ている．まず初期目標に対して，分枝信号線まで多重後方追跡を行い，ステム目標群[2]に追加する．初期目標群が空になったら，ステム目標群から目標を 1 つ取り出し，現在目標群に追加して後方追跡処理を継続する．以上の処理を繰り返し，最初に発見した含意ノードが決定点とする．幅優先探索は，決定点が外部入力に近い場所になる(図 5)．

2.2 正当化

正当化は，未正当化信号線を正当化する操作である．未正当化信号線とは，ゲートの出力信号線値は決定しているが，入力の論理値が X であるゲートの出力信号線である．後方追跡ベースは，前節で述べた含意ノード探索と同様である．一致操作ベース[5]は，未正当化信号線ゲートの入力信号線内で，論理値が X の入力信号線を 1 つ選択し，ゲートの非制御値を割当てる操作である．図 6 に一致操作の例を示す．図 6 中の信号線 h を正当化する．h の入力信号線の内，g=1 を割当て，含意操作を行う．

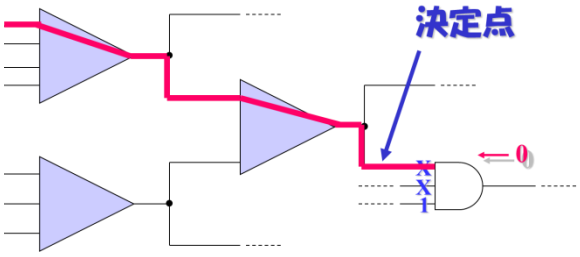


図 4 含意ノード深さ優先探索

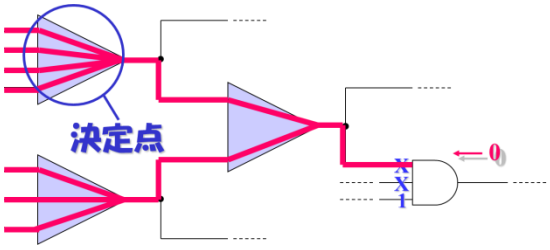


図 5 含意ノード幅優先探索

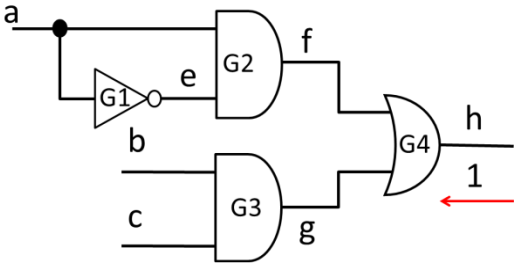


図 6 一致操作例

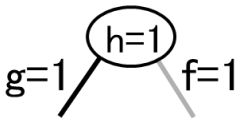


図 7 一致操作決定点

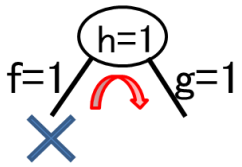


図 8 バックトラック

2.3 バックトラック

バックトラック[1]は，値が矛盾した時，決定点からの割当てを 1 つ戻し，割当てた値と別の値を割当てる操作である．図 7 において f=1 を割当てたとすると，

信号線 a で値の矛盾が発生する．バックトラック操作を行うと，f=1 の割当てと含意操作で割り当てた値をすべて割当て前に戻し，g=1 の割当てを行う．

3. SAT ベーステスト生成

SAT を用いた単一縮退故障のテスト生成法について説明する．例として図 9 の信号線 H が 0 縮退故障を考える．初めにテスト対象となる回路に対して，正常回路(図 9)，単一縮退故障を付加した故障回路（図 10）を CNF に変換する．次に正常回路と故障回路の入力に共通の外部入力を接続する．外部出力に対しては，故障の影響を伝搬，励起させるために片方の外部出力を 1(真)，もう一方を 0(偽)にさせる必要があるため，それぞれの外部出力を EXOR ゲートの入力に接続する．故障影響の観測は少なくとも 1 つの外部出力に伝搬すればよいため，各外部出力を接続した EXOR ゲートの出力を OR ゲートの入力に接続し，OR ゲートの出力が常に 1(真)になる制約を付加する．作成したテスト生成回路モデルの CNF に対して，充足可能性を判定し，充足可能(SAT)と判定されればテスト生成が成功となり，外部入力に割当てた値がテストパターンとなる．また，充足不可能(UNSAT)と判定された場合は対象となる故障に対してテストパターン生成が不可能なため，結果的に冗長故障と判定可能である．

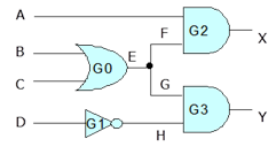
SAT の正当化処理は，どれか 1 つの節でも 1(真)になりえないと判断できた時点で，その割当ての組合せが正当化不可能と判定する．したがって，効率的にかつ早期に割当ての矛盾を発見でき，高速な正当化判定処理が可能である．

4. 実験結果

本実験では，経路活性化法と SAT ベーステスト生成の計 6 種類のテスト生成アルゴリズムを ITC'99 ベンチマーク回路に対して実験を行った．経路活性化法のアルゴリズムでは静的学習を用いている。

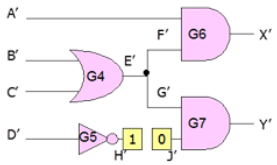
表 1 に各テスト生成アルゴリズムでテスト生成実行時の検出困難故障数を示す．ここで検出困難故障とは，テスト生成に 1 秒以上かかる故障を対象とする．また冗長故障は含まないものとする．表中の(1)は故障伝搬にパスセグメント活性化，正当化に含意深さ優先探索を用いたテスト生成アルゴリズムを示す．また，表中の(3)は故障伝搬に後方追跡，後方追跡と正当化に含意房左右船探索を用いたテスト生成アルゴリズムを示す．表 1 から，(1)が b14_C，b20_C，b21_C で検出困難故障数が最も少なく，(6)が b15_C，b17_C，b22_C で検出困難故障数が最も少ないことがわかる．

表 2 に各テスト生成アルゴリズムで検出困難故障の積集合をとった結果，共通する検出困難故障数を示す．



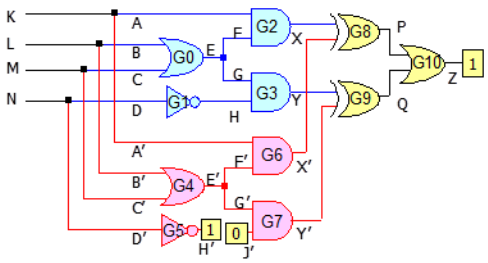
$(\overline{X}+A)\cdot(X+\overline{F})\cdot(\overline{A}+\overline{F}+Z)\cdot(E+\overline{B})\cdot(E+\overline{C})\cdot(B+C+\overline{E})\cdot(D+H)\cdot(\overline{D}+\overline{H})$
$\cdot(\overline{Y}+G)\cdot(Y+\overline{H})\cdot(\overline{G}+\overline{H}+Y)\cdot(\overline{E}+F)\cdot(E+\overline{F})\cdot(\overline{E}+G)\cdot(E+\overline{G})$

図 9 正常回路 CNF



$(\overline{X'}+A')\cdot(X'+\overline{F'})\cdot(\overline{A'}+\overline{F'}+Z')\cdot(E'+\overline{B'})\cdot(E'+\overline{C'})\cdot(B'+C'+\overline{E'})\cdot(D'+H')\cdot(\overline{D'}+\overline{H'})$
$\cdot(\overline{Y'}+G')\cdot(Y'+\overline{J'})\cdot(\overline{G'}+\overline{J'}+Y')\cdot(\overline{E'}+F')\cdot(E'+\overline{F'})\cdot(\overline{E'}+G')\cdot(E'+\overline{G'})\cdot(H')\cdot(\overline{J'})$

図 10 故障回路 CNF



$(\overline{X}+A)\cdot(X+\overline{F})\cdot(\overline{A}+\overline{F}+Z)\cdot(E+\overline{B})\cdot(E+\overline{C})\cdot(B+C+\overline{E})\cdot(D+H)\cdot(\overline{D}+\overline{H})$
$\cdot(\overline{Y}+G)\cdot(Y+\overline{H})\cdot(\overline{G}+\overline{H}+Y)\cdot(\overline{E}+F)\cdot(E+\overline{F})\cdot(\overline{E}+G)\cdot(E+\overline{G})$
$\cdot(\overline{X'}+A')\cdot(X'+\overline{F'})\cdot(\overline{A'}+\overline{F'}+Z')\cdot(E'+\overline{B'})\cdot(E'+\overline{C'})\cdot(B'+C'+\overline{E'})\cdot(D'+H')\cdot(\overline{D'}+\overline{H'})$
$\cdot(\overline{Y'}+G')\cdot(Y'+\overline{J'})\cdot(\overline{G'}+\overline{J'}+Y')\cdot(\overline{E'}+F')\cdot(E'+\overline{F'})\cdot(\overline{E'}+G')\cdot(E'+\overline{G'})\cdot(H')\cdot(\overline{J'})$
$\cdot(\overline{K}+A)\cdot(K+\overline{A})\cdot(\overline{K}+A')\cdot(K+\overline{A'})\cdot(\overline{L}+B)\cdot(L+\overline{B})\cdot(\overline{L}+B')\cdot(L+\overline{B'})$
$\cdot(\overline{M}+C)\cdot(M+\overline{C})\cdot(\overline{M}+C')\cdot(M+\overline{C'})\cdot(\overline{N}+D)\cdot(N+\overline{D})\cdot(\overline{N}+D')\cdot(N+\overline{D'})$
$\cdot(\overline{X}+X+P)\cdot(X+\overline{X}+P)\cdot(\overline{X}+\overline{X}+\overline{P})\cdot(X+X+\overline{P})\cdot(\overline{Y}+Y+Q)\cdot(Y+\overline{Y}+Q)\cdot(\overline{Y}+\overline{Y}+\overline{Q})\cdot(Y+Y+\overline{Q})$
$\cdot(Z+\overline{P})\cdot(Z+\overline{Q})\cdot(P+Q+\overline{Z})\cdot(Z)$

図 11 テスト生成回路モデル

表中の(1)-(6)は 6 つのテスト生成アルゴリズムの検出困難故障で積集合をとった結果を示す．表 2 から，経路活性化法と SAT ベーステスト生成では b17_C を除き，共通の検出困難故障数は 0 となった．

表 3 に経路活性化法の各テスト生成アルゴリズムと SAT ベーステスト生成アルゴリズムの共通の検出困難故障数を示す．表から(1)と(6)のアルゴリズムを組合せることで，b17_C 以外のすべての回路の故障が 1 秒以内で検出可能とわかる．

図 12 は(1)のテスト生成アルゴリズムを用いて，b20_C のテスト生成を行った時のゲート段数と CPU 時間を示す．同様に図 13 は(6)のテスト生成アルゴリズムを用いた場合を示す．図 12，13 から(6)は(1)と比較し，外部入力に近い故障に対して，テスト生成時間がかかることがわかる．

5. おわりに

本稿では，テスト生成アルゴリズムを実装し，その検出困難故障数について評価を行った．結果より(1)と(6)のテスト生成アルゴリズムを組合せることで検出困難故障を大幅に減らすことができるとわかった．今後，テスト生成における決定点の違いなど詳細なテスト生成情報の解析を行う．

表 1 検出困難故障数

		パスセグメント活性化			後方追跡		SAT
		含意深さ	含意幅	一致操作	含意深さ	含意幅	
回路名	対象故障数	(1)	(2)	(3)	(4)	(5)	(6)
b14_C	22630	16	24	123	32	32	122
b15_C	21988	83	193	123	947	914	23
b17_C	76625	1004	1738	979	1958	3135	425
b20_C	45259	42	48	1272	1050	220	451
b21_C	46154	26	50	1286	1088	288	483
b22_C	67536	262	329	1518	1207	641	121

表 2 検出困難故障積集合

	積集合			
回路名	(1)(2)(3)	(4)(5)	(1)-(5)	(1)-(6)
b14_C	11	11	1	0
b15_C	38	547	8	0
b17_C	450	1382	66	2
b20_C	13	61	2	0
b21_C	16	57	4	0
b22_C	57	282	12	0

表 3 経路活性化法と SAT の積集合

回路名	テスト生成アルゴリズム	(1)	(2)	(3)	(4)	(5)
b14_C	(6)	0	0	0	0	0
b15_C		0	1	0	7	9
b17_C		42	39	38	41	92
b20_C		0	0	0	5	0
b21_C		0	0	0	5	1
b22_C		0	0	0	5	0

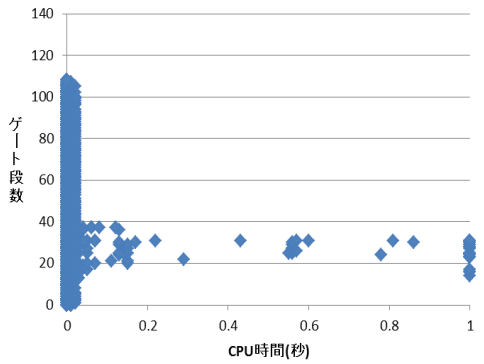


図 12 (1)のゲート段数と CPU 時間

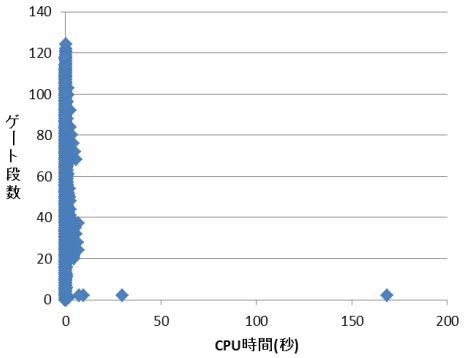


図 13 (6)のゲート段数と CPU 時間

参考文献

[1] MIRON Abramovici , Melvin A.Breuer , Arthur D.Friedman “DIGITAL SYSTEMS TESTING AND TESTABLE DESIGN”, 1995.

[2] H. Fujiwara, T. Shimono, “On the Acceleration of Test Generation Algorithms”, IEEE Transactions on Electronic Computers, Vol.32, No.12, Dec , 1983, pp. 1137-1144.

[3] MICHAEL H. SCHULZ, ERWIN TRISCHLER, THOMAS M. SAPFERT, “SOCRATES: a highly efficient automatic test pattern generation system”, IEEE Transactions on Computer-Aided Design, Vol.7, No.1, Jan , 1988, pp.126-137

[4] M. Henftling, H. C. Wittmann, K. J. Antreich, “A Single-Path-Oriented Fault-Effect Propagation in Digital Circuits”, IEEE/ACM International Conference, Nov, 1995, pp.304-309.

[5] J. PAUL ROTH, WILLARD G. BOURICIUS, PETER R, Programmed Algorithms to Compute Tests to Detect and Distinguish Between Failures in Logic Circuits, Electronic Computers, IEEE Transactions , Oct. 1967, p567-580.

[6] Tracy Larrabee, “Test Pattern Generation Using Boolean Satisfiability” , IEEE TRANSACTION ON COMPUTER-AIDEDDESIGN, VOL.11, No.1 , 1992.

[7] Mituo Teramoto, ” A Method for Reducing the Search Space in Test Pattern Generation, “ Proc. International Test Conference, 1993, pp. 429-435.